DYNIX/ptx Reference Manual
Section 2

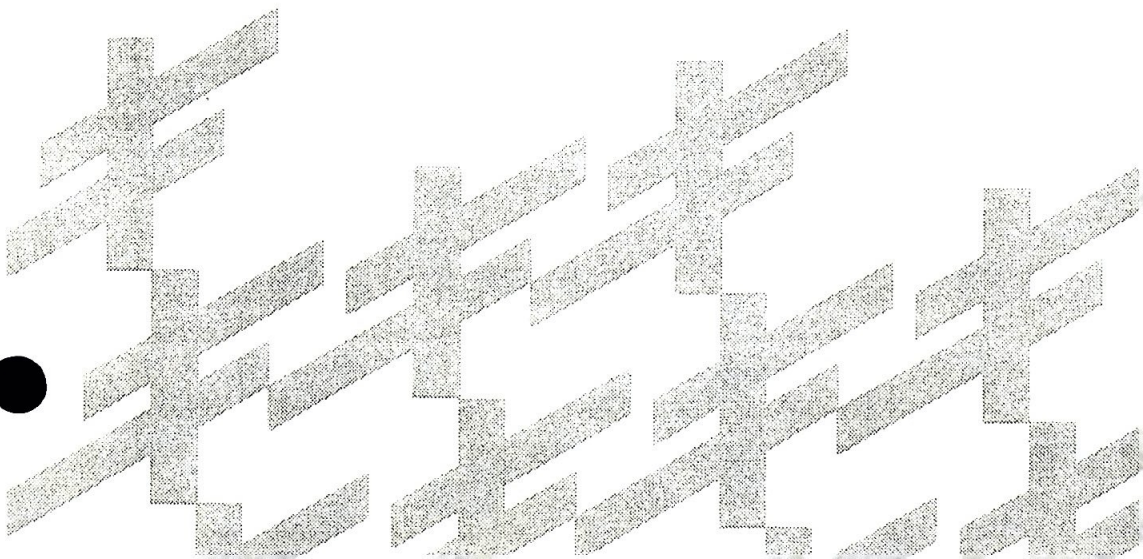
1003-49232-01

THIS SHEET LIFTER
MUST BE PLACED ON
TOP OF CONTENTS

GRAPHIC PRODUCTS
Chicago, Illinois 60609

DYNIX/ptx®
Reference Manual
Section 2

1003-49232-01
Sequent Computer Systems, Inc.



ACT is a trademark of Micro-Term.

Ann Arbor is a trademark of Ann Arbor Terminals.

Beehive is a trademark of Beehive International.

Concept is a trademark of Human Designed Systems.

Dataphone is a registered trademark of AT&T.

DEC is a trademark of Digital Equipment.

Diablo is a registered trademark of Xerox.

Digital is a trademark of Digital Equipment.

DOCUMENTER'S WORKBENCH is a trademark of AT&T.

DYNIX and DYNIX/ptx are registered trademarks of Sequent Computer Systems, Inc.

HP is a trademark of Hewlett-Packard.

i386 is a trademark of Intel Corp.

LSI is a trademark of Lear Siegler.

Micro-term and MIME are trademarks of Micro-Term.

MULTIBUS is a trademark of Intel Corp.

PDP is a trademark of Digital Equipment.

Sequent is a trademark of Sequent Computer Systems, Inc.

Symmetry is a registered trademark of Sequent Computer Systems, Inc.

Tektronix and Tektronix 4010 are registered trademarks of Tektronix.

Teleray is a trademark of Research.

Teletype is a registered trademark of AT&T.

TeleVideo is a registered trademark of TeleVideo Systems.

UNIX is a registered trademark of AT&T.

VAX is a trademark of Digital Equipment.

Versatec is a registered trademark of Versatec.

VT100 is a trademark of Digital Equipment.

Copyright © 1989, 1990 by Sequent Computer Systems, Inc. All rights reserved. This document may not be copied or reproduced in any form without permission from Sequent Computer Systems, Inc. Information in this document is subject to change without notice.

Printed in the United States of America.

About This Manual

Overview

The *DYNIX/ptx Reference Manual* is for programmers working in the DYNIX/ptx operating system environment. This manual describes the programming features of the operating system and assumes that the audience is familiar with the operating system. It does not provide a general overview of the operating system or implementation details of the system.

Organization

The *DYNIX/ptx Reference Manual* is contained in five binders organized as follows:

- Section 1 A-F **Commands**
- Section 1 G-Z **Commands**
- Section 2 **System Calls**
  Section 2SEQ **Sequent System Calls**
  Section 3 **Subroutines**
    - Section 3C **C Library Routines**
    - Section 3M **Mathematical Library Routines**
    - Section 3N **Networking Support Utilities**
    - Section 3PPS **Sequent Parallel Programming Library**
    - Section 3S **Standard I/O Library Routines**
    - Section 3SEQ **Sequent I/O Library Routines**
    - Section 3X **Special Routines**
- Section 4 **File Formats**
  Section 5 **Miscellaneous Facilities**
- Section 7 **Special Files**
  Section 8 **Stand-alone Utilities**

Each binder also contains the following items:

- **Table of Contents** that alphabetically lists the contents of the entire manual set
- **Permuted Index** that lists each programming feature in the entire manual set

The permuted index is a list of keywords, given alphabetically in the second of three columns, together with the context in which each keyword is found. The following shows four separate entries illustrating four ways that the **cat** command can be looked up in the permuted index. Note that you can find the **cat** command by looking up "cat," "concatenate," "files," or "print."

```
                    cat: concatenate and print files.....cat(1)
               cat: concatenate and print files..... cat(1)
concatenate and print files.....cat(1)
     concatenate and print files..... cat(1)
```

Keywords are either topical keywords or the names of manual entries themselves. Entries are identified with their section numbers shown in parentheses. The numbers are important because there is duplication of names among the sections. The right column lists the name of the manual page on which each keyword may be found. The left column contains useful information about the keyword.

The following paragraphs briefly describe the other sections of the manual.

Section 1: Commands describes operating system commands and commands that support C and other programming languages.

Section 2: System Calls describes the access to the services provided by the operating system kernel, including the C language interface.

Section 2SEQ: Sequent System Calls describes the access to the Sequent services provided by the operating system kernel, including the C language interface.

Section 3: Subroutines describes the available subroutines. Their binary versions reside in various system libraries in the directories */lib* and */usr/lib*. The introduction to Section 3 describes these libraries and the files in which they are stored.

Section 4: File Formats documents the structure of particular kinds of files; for example, the format of the output of the link editor is given in *a.out*(4). Excluded are files used by only one command (for example, the assembler's intermediate files). In general, the C language structures corresponding to these formats can be found in the directories */usr/include* and */usr/include/sys*.

Section 5: Miscellaneous Facilities contains a variety of descriptions including descriptions of character sets, macro packages, and so forth.

Section 7: Special Files describes various special files that refer to specific hardware peripherals and operating system device drivers.

Section 8: Stand-alone Utilities describes stand-alone procedures.

References to manual pages with numbers other than those above mean that the utility is contained in the appropriate section of another manual. Each section consists of independent entries which are alphabetized, with the exception of the introductions that begin some sections. Section 3 is divided into subsections in alphabetical order by suffix. Some entries may describe several routines, commands, and so forth. In such cases, the entry appears only once, alphabetized under its *primary* name, the name that appears at the upper corners of each manual page.

Each manual page can be viewed online with the **man**(1) command. For example, enter **man shell** to view the **shell** manual page. Enter **man man** for more information about the **man** command.

Notational Conventions

Manual pages are based on a common format. The main headings are described here—not all of them appear in all cases:

- **NAME** gives the name(s) of the entry and briefly states its purpose. This entry is used to create the Permuted Index and the Table of Contents.
- **SYNOPSIS** summarizes the usage of the program being described. Conventions vary in some sections but are generally as follows:

- **Boldface** strings are literals and are to be typed just as they appear.
- *Italic* strings usually represent substitutable argument prototypes and program names found elsewhere in the manual.
- Square brackets [ ] around an argument prototype indicate that the argument is optional. When an argument prototype is given as *name* or *file*, it refers to a *filename*.
- Ellipses ... are used to show that the previous argument prototype may be repeated.
- A final convention is used by the commands themselves. An argument beginning with a minus –, plus +, or equal sign = is often taken to be some sort of flag argument even if it appears in a position where a file name could appear. Therefore, it is unwise to have files whose names begin with –, +, or =.

- **CONFIGURATION ENTRY** shows how a driver is called out in a system configuration file.
- **DESCRIPTION** describes the entry, including its purpose and how it is used.
- **FIELDS** describe system file or record entries.
- **EXAMPLES** and **BINARY CONFIGURATION** give examples of usage.
- **FILES** list files that are part of a program, created by a program, or related to a program.
- **SEE ALSO** points to related information in other manual pages.
- **DIAGNOSTICS** discuss the diagnostic (error) messages that may be produced. Messages that are self-explanatory are not listed.
- **WARNINGS** contains warnings against improper user actions.
- **NOTES** give hints about the use of the utility.
- **BUGS** describe known bugs in the software.

Contents

1 Commands

intro(1) ..... introduction to maintenance commands and application programs
300(1): 300, 300s ..... handle special functions of DASI 300 and 300s terminals
4014(1) ..... paginator for the Tektronix 4014 terminal
450(1) ..... handle special functions of the DASI 450 terminal
accept(1M): accept, reject ..... allow or prevent LP requests
acct(1M): acctdisk, acctdusg, accton, acctwtmp ..... overview of accounting and miscellaneous accounting commands
acctcms(1M) ..... command summary from per-process accounting records
acctcom(1) ..... search and print process accounting files
acctcon(1M): acctcon1, acctcon2 ..... connect-time accounting
acctmerg(1M) ..... merge or add total accounting files
acctprc(1M): acctprc1, acctprc2 ..... process accounting
acctsh(1M): chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup, turnacct ..... shell procedures for accounting
addcmdopt(1M) ..... add a command option to a menu
addformopt(1M) ..... add a form/SELECT window option to a menu
addmenuopt(1M) ..... add a menu option to a menu
admin(1) ..... create and administer SCCS files
apropos(1) ..... locate commands by keyword lookup
ar(1) ..... see *C and Language Tools Binder*
as(1) ..... see *C and Language Tools Binder*
asa(1) ..... interpret ASA carriage control characters
assist(1) ..... assistance using operating system commands
astgen(1) ..... program for generating/modifying ASSIST menus or command forms
at(1): at, batch ..... execute commands at a later time
awk(1) ..... pattern scanning and processing language
banner(1) ..... make posters
basename(1): basename, dirname ..... deliver portions of pathnames
bc(1) ..... arbitrary-precision arithmetic language
bcheckrc(1M) ..... system initialization procedures
bdflush(1M) ..... update daemon
bdiff(1) ..... big diff
bfs(1) ..... big file scanner
bomverify(1M) ..... perform an installation consistency check based on the BOM
bootflags(1M) ..... display/change system boot flags
bp(1M) ..... binary patcher
cal(1) ..... print calendar

calendar(1) ..... reminder service
captoinfo(1M) ..... convert a termcap description into a terminfo description
cat(1) ..... concatenate and print files
cb(1) ..... see *C and Language Tools Binder*
cc(1) ..... see *C and Language Tools Binder*
cd(1) ..... change working directory
cdc(1) ..... change the delta commentary of an SCCS delta
cflow(1) ..... see *C and Language Tools Binder*
checkeq(1) ..... check documents for correct eqn syntax.
checkfsys(1M) ..... check a filesystem
chmod(1) ..... change mode
chown(1): chown, chgrp ..... change owner or group
chroot(1M) ..... change root directory for a command
chrtbl(1M) ..... generate character classification and conversion tables
ci(1) ..... check in RCS revisions
ckbupscd(1M) ..... check filesystem backup schedule
clri(1M) ..... clear inode
cmp(1) ..... compare two files
co(1) ..... check out RCS revisions
col(1) ..... filter reverse line-feeds
comb(1) ..... combine SCCS deltas
comm(1) ..... select or reject lines common to two sorted files
compress(1): compress, uncompress, zcat ..... compress and expand data
config(1M) ..... build system configuration files
coproc(1F): cocreate, cosend, cocheck, coreceive, codestroy ..... communicate to a process
cp(1): cp, ln, mv ..... copy, link or move files
cpp(1) ..... see *C and Language Tools Binder*
cpio(1) ..... copy file archives in and out
cprs(1) ..... see *C and Language Tools Binder*
cpset(1) ..... copy file and set mode, user, and group IDs
crash(1M) ..... examine system images
cron(1M) ..... clock daemon
crontab(1) ..... user crontab file
crypt(1) ..... encode/decode
csh(1) ..... a shell (command interpreter) with C-like syntax
csplit(1) ..... context split
ct(1C) ..... spawn getty to a remote terminal
ctags(1) ..... create a tags file
ctrace(1) ..... see *C and Language Tools Binder*
cu(1C) ..... call another UNIX system
cut(1) ..... cut out selected fields of each line of a file
cxref(1) ..... see *C and Language Tools Binder*
date(1) ..... print and set the date
dbx(1) ..... see *C and Language Tools Binder*
dc(1) ..... desk calculator
dcheck(1M) ..... filesystem directory consistency check
dd(1M) ..... convert and copy a file
delta(1) ..... make a delta (change) to an SCCS file

deroff(1) ..... remove nroff, tbl, and eqn constructs
devbuild(1M): devbuild,
devdestroy ..... build and destroy virtual devices
devnm(1M) ..... device name
df(1M) ..... report number of free disk blocks and inodes
diff(1) ..... differential file comparison
diff3(1) ..... 3-way differential file comparison
dircmp(1) ..... directory comparison
dis(1) ..... see *C and Language Tools Binder*
diskusg(1M) ..... generate disk accounting data by user ID
dltsectbl(1M) ..... refresh security table
dmesg(1M) ..... collect system diagnostic messages to form error log
du(1M) ..... summarize disk usage
dump(1) ..... see *C and Language Tools Binder*
dump(1M) ..... incremental filesystem dump
dumpconf(1M) ..... dump kernel auto-configuration device table
dumpfs(1M) ..... dump filesystem information
ec(1M) ..... forms compiler
echo(1) ..... echo arguments
echo(1F) ..... put string on virtual output
ed(1): ed, red ..... text editor
edc(1M) ..... run an externally described command
edit(1) ..... text editor (variant of ex for casual users)
edquota(1M) ..... edit user quotas
efl(1) ..... Extended Fortran Language
egrep(1) ..... search a file for a pattern using full regular expressions
enable(1): enable, disable ..... enable/disable LP printers
env(1) ..... set environment for command execution
ex(1) ..... text editor
expr(1) ..... evaluate arguments as an expression
factor(1) ..... obtain the prime factors of a number
ff(1M) ..... list filenames and statistics for a filesystem
fgrep(1) ..... search a file for a character string
file(1) ..... determine file type
find(1) ..... find files
format(1M) ..... format disks or perform other defect management functions
fsck(1M) ..... filesystem consistency check and interactive repair
fsirand(1M) ..... install random inode generation numbers
fsplit(1) ..... split f77, ratfor, or efl files
fsstat(1M) ..... report filesystem status
fstyp(1M) ..... determine filesystem identifier
fuser(1M) ..... identify processes using a file or file structure
fwtmp(1M): fwtmp, wtmpfix ..... manipulate connect accounting records
gdev(1G): hpd, erase, hardcopy,
tekset, td ..... graphical device routines and filters
ged(1G) ..... graphical editor
gencc(1M) ..... create a front-end to the cc command
get(1) ..... get a version of an SCCS file
getfrm(1F) ..... returns the current frame number

getitems(1F) ..... returns a list of the currently marked menu items
getopt(1) ..... parse command options
getopts(1): getopts, getoptcvt ..... parse command options
getty(1M) ..... set terminal type, modes, speed, and line discipline
glossary(1) ..... definitions of common operating system terms and symbols
graph(1G) ..... draw a graph
graphics(1G) ..... access graphical and numerical commands
greek(1) ..... select terminal filter
grep(1) ..... search a file for a pattern
groups(1) ..... show group memberships
gutil(1G) ..... graphical utilities
help(1) ..... operating system Help Facility
helpadm(1M) ..... change the Help Facility database
hp(1) ..... handle special functions of Hewlett-Packard terminals
hpio(1) ..... Hewlett-Packard 2645A terminal tape file archiver
icheck(1M) ..... filesystem storage consistency check
id(1M) ..... print user and group IDs and names
ident(1) ..... identify files
indicator(1F) ..... display application specific alarms and/or the working indicator
infocmp(1M) ..... compare or print out terminfo descriptions
init(1M): init, telinit ..... process control initialization
install(1M) ..... install commands
installboot(1M) ..... installs a boot program on a disk
ipcrm(1) ..... remove a message queue, semaphore set, or shared memory ID
ipcs(1) ..... report inter-process communication facilities status
isismgr(1M) ..... ISIS file manager, updates text files in ISIS file format
join(1) ..... relational database operator
kill(1) ..... terminate a process
killall(1M) ..... kill all active processes
ksh(1): ksh, rksh ..... the KornShell, a standard/restricted command and programming language
labelit(1M) ..... provide labels for filesystems
ld(1) ..... see *C and Language Tools Binder*
lex(1) ..... generate programs for simple lexical tasks
line(1) ..... read one line
link(1M): link, unlink ..... link and unlink files and directories
lint(1) ..... see *C and Language Tools Binder*
list(1) ..... see *C and Language Tools Binder*
locate(1) ..... use keywords to identify an operating system command
login(1) ..... sign on
logname(1) ..... get login name
lorder(1) ..... see *C and Language Tools Binder*
lp(1): lp, cancel ..... send/cancel requests to the LP print service
lpadmin(1M) ..... configure the LP print service
lpfilter(1M) ..... administer filters used with the LP print service
lpforms(1M) ..... administer forms used with the LP print service

lpsched(1M): lpsched, lpshut,
lpmove ..... start/stop the LP print service and move requests
lpstat(1) ..... print information about the status of the LP print service
lpusers(1M) ..... set printing queue priorities
ls(1) ..... list contents of directory
m4(1) ..... macro processor
machid(1): i386, ns32000, pdp11,
u3b, u3b2, u3b5, vax ..... get processor type truth value
mail(1): mail, rmail ..... send mail to users or read mail
mailbug(1) ..... submit a bug report or enhancement request
mailx(1) ..... interactive message processing system
make(1) ..... maintain, update, and regenerate groups of programs
makealiases(1M): makealiases,
makewhatis ..... generate man page databases
makedev(1M) ..... make system special files
makekey(1) ..... generate encryption key
man(1) ..... display on-line man pages
mcs(1) ..... manipulate the object file comment section
menu(1M) ..... menu processor
menumsg(1M) ..... route user message to menu message queue
merge(1) ..... three-way file merge
mesg(1) ..... permit or deny messages
message(1F) ..... puts its arguments on message line
mkdir(1) ..... make directories
mkfs(1M) ..... construct a filesystem
mknod(1M) ..... build special file
mkpwdbm(1M) ..... make a database for fast password lookup
mkvtoc(1M) ..... populate VTOC on disks
monitor(1M) ..... monitor system activity
more(1) ..... view a file one full screen at a time
mount(1M): mount, umount ..... mount and unmount filesystems
mountall(1M): mountall,
umountall ..... mount, unmount multiple filesystems
mt(1) ..... magnetic tape manipulating program
mvdir(1M) ..... move a directory
nawk(1) ..... pattern scanning and processing language
ncheck(1M) ..... generate pathnames from i-numbers
newform(1) ..... change the format of a text file
newfs(1M) ..... construct a new file system
newgrp(1M) ..... log in to a new group
news(1) ..... print news items
nice(1) ..... run a command at low priority
nl(1) ..... line numbering filter
nm(1) ..... see *C and Language Tools Binder*
noage(1M): noage, nopff, noswap,
onproc ..... run a command with priority aging, PFF, swapping, or processor migration disabled
nohup(1) ..... run a command immune to hangups and quits
od(1) ..... octal dump
online(1M): online, offline ..... bring processors online and offline

pack(1): pack, pcat, unpack ..... compress and expand files
passmgmt(1M) ..... password files management
passwd(1) ..... change login password and password attributes
paste(1) ..... merge same lines of several files or subsequent lines of one file
pathconv(1F) ..... search Interpreter criteria for filename
pdbx(1) ..... see *C and Language Tools Binder*
pg(1) ..... file perusal filter for CRTs
pmap(1M) ..... display or alter parameters of virtual-to-physical mapping devices
pr(1) ..... print files
prof(1) ..... display profile data
prompt(1M) ..... prompt program
prs(1) ..... print an SCCS file
prtvtoc(1M) ..... print the VTOC of a block device
ps(1) ..... report process status
pwck(1M): pwck, grpck ..... password/group file checkers
pwconv(1M) ..... installs and updates /etc/shadow with information from /etc/passwd
pwd(1) ..... working directory name
pwdhd(1M) ..... daemon to oversee password database creation
pwfix(1M) ..... remove incorrectly formatted password entries
pwunconv(1M) ..... converts from a two to a one password file scheme
quota(1) ..... display disk usage and limits
quotacheck(1M) ..... filesystem quota consistency checker
quotaon(1M): quotaon, quotaoff ..... turn filesystem quotas on and off
ratfor(1) ..... rational Fortran dialect
rc0(1M) ..... run commands performed to stop the operating system
rc2(1M) ..... run commands performed for multi-user environment
rcs(1) ..... change RCS file attributes
rcsdiff(1) ..... compare RCS revisions
rcsintro(1) ..... introduction to RCS commands
rcsmerge(1) ..... merge RCS revisions
readfile(1F): readfile, longline ..... reads file and gets longest line
regcmp(1) ..... regular expression compile
regex(1F) ..... match patterns against a string
reinit(1F) ..... run an initialization file
repquota(1M) ..... summarize quotas for a filesystem
reset(1F) ..... reset the current form field to its default values
restore(1M) ..... incremental filesystem restore
rlog(1) ..... print log messages and other information about RCS files
rm(1): rm, rmdir ..... remove files or directories
rmdel(1) ..... remove a delta from an SCCS file
rmvopt(1M) ..... remove a menu option
run(1F) ..... run an executable
runacct(1M) ..... run daily accounting
sact(1) ..... print current SCCS file editing activity
sag(1G) ..... system activity graph
sar(1) ..... system activity reporter
sar(1M): sa1, sa2, sadc ..... system activity report package

savecore(1M) ..... save a core dump of the operating system
sccsdiff(1) ..... compare two versions of an SCCS file
sccstorcs(1) ..... build RCS file from SCCS file
sdiff(1) ..... side-by-side difference program
sed(1) ..... stream editor
set(1F): set, unset ..... set and unset environment variables in core or in files
setcolor(1F) ..... redefine or create a color
setmnt(1M) ..... establish mount table
setpgrp(1) ..... run command in its own process group
setup(1) ..... initialize system for first user
sh(1): sh, rsh ..... shell, the standard/restricted command programming language
shell(1F) ..... run a command using shell
showcfg(1M) ..... show the configuration of the machine
shutdown(1M) ..... shut down system, change system state
size(1) ..... see *C and Language Tools Binder*
sleep(1) ..... suspend execution for an interval
sort(1) ..... sort and/or merge files
spell(1): spell, hashmake, spellin, hashcheck ..... find spelling errors
spline(1G) ..... interpolate smooth curve
split(1) ..... split a file into pieces
starter(1) ..... information about the operating system for beginning users
stat(1G) ..... statistical network useful with graphical commands
strace(1M) ..... print STREAMS trace messages
strclean(1M) ..... STREAMS error logger cleanup program
strerr(1M) ..... STREAMS error logger daemon
strip(1) ..... see *C and Language Tools Binder*
stty(1) ..... set the options for a terminal
su(1M) ..... become superuser or another user
sum(1) ..... print checksum and block count of a file
swap(1M) ..... swap administrative interface
sweepon(1M): sweepon, sweepoff ..... enable/disable patrol seeks on DCC disks
sync(1M) ..... update the superblock
sysadm(1) ..... menu interface to do system administration
tabs(1) ..... set tabs on a terminal
tail(1) ..... deliver the last part of a file
tar(1) ..... tape file archiver
tasktbl(1M) ..... create security table in shared memory
tee(1) ..... pipe fitting
test(1) ..... condition evaluation command
tfipc(1M) ..... interprocess communication reporter
tfmenush(1M) ..... assign menu system as user login shell
tfnotify(1M) ..... send completion message
tic(1M) ..... terminfo compiler
time(1): time, ptime ..... time a command
timex(1) ..... time a command; report process data and system activity
toc(1G): dtoc, ttoc, vtoc ..... graphical table of contents routines
touch(1) ..... update access and modification times of a file

tplot(1G) ..... graphics filters
tput(1) ..... initialize a terminal or query terminfo database
tr(1) ..... translate characters
true(1): true, false ..... provide truth values
tsort(1) ..... topological sort
tty(1) ..... get the name of the terminal
ttydevname(1) ..... get the device associated with a terminal or the terminal associated with a device
tunefs(1M) ..... tune up an existing filesystem
uadmin(1M) ..... administrative control
umask(1) ..... set file-creation mode mask
uname(1) ..... print name of current system
unget(1) ..... undo a previous get of an SCCS file
uniq(1) ..... report repeated lines in a file
units(1) ..... conversion program
uptime(1) ..... show how long system has been up
usage(1) ..... retrieve a command description and usage examples
uucheck(1M) ..... check the uucp directories and permissions file
uucico(1M) ..... file transport program for the uucp system
uucleanup(1M) ..... uucp spool directory clean-up
uucp(1C): uucp, uulog, uuname ..... UNIX-to-UNIX system copy
uugetty(1M) ..... set terminal type, modes, speed, and line discipline
uusched(1M) ..... the scheduler for the uucp file transport program
uustat(1C) ..... uucp status inquiry and job control
uuto(1C): uuto, uupick ..... public UNIX-to-UNIX system file copy
uutry(1M) ..... try to contact remote system with debugging on
uux(1C) ..... UNIX-to-UNIX system command execution
uuxqt(1M) ..... execute remote command requests
val(1) ..... validate SCCS file
vc(1) ..... version control
vi(1) ..... screen-oriented (visual) display editor based on ex
vmtune(1M) ..... tune system-wide virtual-memory parameters
volcopy(1M) ..... make literal copy of filesystem
vsig(1F) ..... synchronize a co-process with the controlling FMLI object
wait(1) ..... await completion of process
wall(1) ..... write to all users
wc(1) ..... word count
what(1) ..... identify SCCS files
whatis(1) ..... describe what a command is
who(1) ..... who is on the system
whodo(1M) ..... who is doing what
write(1) ..... write to another user
xargs(1) ..... construct argument list(s) and execute command
yacc(1) ..... yet another compiler-compiler

2 System Calls

intro(2) ..... introduction to system calls and error numbers
access(2) ..... determine accessibility of a file
acct(2) ..... enable or disable process accounting
alarm(2) ..... set a process alarm clock
brk(2): brk, sbrk ..... change data segment space allocation
chdir(2) ..... change working directory
chmod(2) ..... change mode of file
chown(2) ..... change owner and group of a file
chroot(2) ..... change root directory
close(2) ..... close a file descriptor
creat(2) ..... create a new file or rewrite an existing one
dup(2) ..... duplicate an open file descriptor
dup2(2) ..... duplicate an open file descriptor
exec(2): execl, execv, execle,
execve, execlp, execvp ..... execute a file
exit(2): exit, _exit ..... terminate process
fcntl(2) ..... file control
fork(2) ..... create a new process
getdents(2) ..... read directory entries and put in a file system independent format
getgroups(2) ..... get supplementary group IDs
getmsg(2) ..... get next message off a stream
getpid(2): getpid, getpgrp, getppid .... get process, process group, and parent process IDs
getuid(2): getuid, geteuid, getgid,
getegid ..... get real user, effective user, real group, and effective group IDs
ioctl(2) ..... control device
kill(2) ..... send a signal to a process or a group of processes
link(2) ..... link to a file
lseek(2) ..... move read/write file pointer
mkdir(2) ..... make a directory
mknod(2): mknod ..... make a directory, or a special or ordinary file
mount(2) ..... mount a filesystem
msgctl(2) ..... message control operations
msgget(2) ..... get message queue
msgop(2) ..... message operations
nice(2) ..... change priority of a process
open(2) ..... open for reading or writing
pathconf(2): pathconf, fpathconf ..... get configurable pathname variables
pause(2) ..... suspend process until signal
pipe(2) ..... create an interprocess channel
plock(2) ..... lock process, text, or data in memory
poll(2) ..... STREAMS input/output multiplexing
profil(2) ..... execution time profile
ptrace(2): ptrace, mptrace ..... process trace facility
putmsg(2) ..... send a message on a stream
read(2) ..... read from file
rename(2) ..... change the name of a file
rmdir(2) ..... remove a directory

semctl(2) ..... semaphore control operations
semget(2) ..... get set of semaphores
semop(2) ..... semaphore operations
setpgid(2) ..... manipulate signal sets
setpgrp(2) ..... set process group ID
setsid(2) ..... create session and set process group ID
setuid(2): setuid, setgid ..... set user and group IDs
shmctl(2) ..... shared memory control operations
shmget(2) ..... get shared memory segment identifier
shmop(2): shmop, shmat, shmdt ..... shared memory operations
sigaction(2) ..... examine and change signal actions
sigemptyset(2): sigemptyset,
sigfillset, sigaddset, sigdelset,
sigismember ..... manipulate signal sets
signal(2) ..... specify what to do upon receipt of a signal
sigpending(2) ..... examine pending signals
sigprocmask(2) ..... examine and change blocked signals
sigset(2): sigset, sighold, sigrelse,
sigignore, sigpause ..... signal management
sigsuspend(2) ..... wait for a signal
stat(2): stat, lstat, fstat ..... get file status
statfs(2): statfs, fstatfs ..... get filesystem information
stime(2) ..... set time
stty(2): stty, gtty ..... set and get terminal state (defunct)
sync(2) ..... update superblock
syscall(2) ..... indirect system call
sysconf(2) ..... get configurable system variables
sysi86(2) ..... machine specific functions
tcgetattr(2): tcgetattr, tcsetattr ..... get/set terminal state
tcgetpgrp(2) ..... get foreground process group ID
tcsetpgrp(2) ..... set foreground process group ID
time(2) ..... get time
times(2) ..... get process and child process times
uadmin(2) ..... administrative control
ulimit(2) ..... get and set user limits
umask(2) ..... set and get file creation mask
umount(2) ..... unmount a filesystem
uname(2) ..... get name of current operating system
unlink(2) ..... remove directory entry
ustat(2) ..... get filesystem statistics
utime(2) ..... set file access and modification times
vfork(2) ..... spawn new process in a virtual memory efficient way
wait(2): wait, waitpid ..... wait for a child process to stop or terminate
write(2) ..... write on a file

2SEQ Sequent System Calls

intro(2SEQ) ..... introduction to Sequent system calls
bdflush(2SEQ) ..... update daemon
fvhangup(2SEQ) ..... virtually hangup the specified controlling terminal
get_process_stats(2SEQ) ..... get resource utilization information
getdtablemax(2SEQ) ..... get maximum in-use descriptor table index
getdtablesize(2SEQ) ..... get descriptor table size
getpagesize(2SEQ) ..... get system page size
mmap(2SEQ) ..... map an open file into the processs address space
munmap(2SEQ) ..... unmap part of the processs address space
proc_ctl(2SEQ) ..... control over various process attributes
quotactl(2SEQ) ..... manipulate disk quotas
readlink(2SEQ) ..... read value of a symbolic link
readv(2SEQ) ..... read vector from file
setdtablesize(2SEQ) ..... set descriptor table size
setgroups(2SEQ) ..... set group access list
sigcontext(2SEQ) ..... get signal context
sigstack(2SEQ) ..... set and/or get signal stack context
symlink(2SEQ) ..... make symbolic link to a file
tmp_affinity(2SEQ) ..... bind a process to a processor
tmp_ctl(2SEQ) ..... multi-processor control
truncate(2SEQ): truncate,
ftruncate ..... truncate a file to a specified length
vm_ctl(2SEQ) ..... examine and change virtual memory tuning parameters
writev(2SEQ) ..... write vector on a file

3 Subroutines

intro(3) ..... introduction to functions and libraries

3C C Library Routines

a64l(3C): a64l, l64a ..... convert between long integer and base-64 ASCII string
abort(3C) ..... generate an IOT fault
abs(3C) ..... return integer absolute value
bsearch(3C) ..... binary search a sorted table
clock(3C) ..... report CPU time used
conv(3C): toupper, tolower,
_toupper, _tolower, toascii ..... translate characters
crypt(3C): crypt, setkey, encrypt ..... generate hashing encryption
ctime(3C): ctime, localtime,
gmtime, asctime, cftime, ascftime,
mktime, tzset ..... convert date and time to string
ctype(3C): isdigit, isxdigit,
islower, isupper, isalpha, isalnum,
isspace, iscntrl, ispunct, isprint,
isgraph, isascii, tolower, toupper,
toascci, _tolower, _toupper,
setchrclass ..... character handling

dev_to_ttyname(3C): dev_to_ttyname, ttyname_to_dev ..... convert between major/minor numbers and tty names
dial(3C) ..... establish an out-going terminal line connection
drand48(3C): drand48, erand48, lrand48, nrand48, mrand48, jrand48, srand48, seed48, lcong48 ..... generate uniformly distributed pseudorandom numbers
ecvt(3C): ecvt, fcvt, gcvt ..... convert floating-point number to string
end(3C): end, etext, edata ..... last locations in program
fpgetround(3C): fpgetround, fpsetround, fpgetmask, fpsetmask, fpgetsticky, fpsetsticky ..... IEEE floating-point environment control
frexp(3C): frexp, ldexp, modf ..... manipulate parts of floating-point numbers
ftw(3C) ..... walk a file tree
getcwd(3C) ..... get pathname of current working directory
getenv(3C) ..... return value for environment name
getgrent(3C): getgrent, getgrgid, getgrnam, setgrent, endgrent, fgetgrent ..... get group file entry
getlogin(3C) ..... get login name
getopt(3C) ..... get option letter from argument vector
getpass(3C) ..... read a password
getpw(3C) ..... get name from UID
getpwent(3C): getpwent, getpwuid, getpwnam, setpwent, endpwent, fgetpwent ..... get password file entry
getut(3C): getutent, getutid, getutline, pututline, setutent, endutent, utmpname, getutpid ..... access utmp file entry
hsearch(3C): hsearch, hcreate, hdestroy ..... manage hash search tables
isnan(3C): isnand, isnanf ..... test for floating-point NaN (Not-A-Number)
l3tol(3C): l3tol, ltol3 ..... convert between 3-byte integers and long integers
lockf(3C) ..... record locking on files
lsearch(3C): lsearch, lfind ..... linear search and update
malloc(3C): malloc, free, realloc, calloc ..... main memory allocator
memory(3C): memccpy, memchr, memcmp, memcpy, memset ..... memory operations
mkfifo(3C) ..... make a FIFO special file
mktemp(3C) ..... make a unique filename
monitor(3C) ..... prepare execution profile
nlist(3C) ..... get entries from name list
perror(3C): perror, errno, sys_errlist, sys_nerr ..... system error messages
putenv(3C) ..... change or add value to environment
putpwent(3C) ..... write password file entry
qsort(3C) ..... quicker sort
rand(3C): rand, srand ..... simple random-number generator
setjmp(3C): setjmp, longjmp ..... non-local goto
setlocale(3C) ..... set programs location

sigsetjmp(3C): sigsetjmp,
siglongjmp ..... non-local jumps
sleep(3C) ..... suspend execution for interval
ssignal(3C): ssignal, gsignal ..... software signals
stdipc(3C) ..... standard interprocess communication package
strftime(3C) ..... format time information
string(3C): strcat, strdup, strncat,
strcmp, strncmp, strcpy, strncpy,
strlen, strchr, strrchr, strpbrk,
strspn, strcspn, strtok ..... string operations
strstr(3C) ..... find first occurrence of character
strtod(3C): strtod, atof ..... convert string to double-precision number
strtol(3C): strtol, atol, atoi ..... convert string to integer
swab(3C) ..... swap bytes
tsearch(3C): tsearch, tfind,
tdelete, twalk ..... manage binary search trees
ttyname(3C): ttyname, isatty ..... find name of a terminal
ttyslot(3C) ..... find the slot in the utmp file of the current user

3M Mathematical Library Routines

bessel(3M): j0, j1, jn, y0, y1, yn ..... Bessel functions
erf(3M): erf, erfc ..... error function and complementary error function
exp(3M): exp, log, log10, pow, sqrt .... exponential, logarithm, power, square root functions
floor(3M): floor, ceil, fmod, fabs ..... floor, ceiling, remainder, absolute value functions
gamma(3M) ..... log gamma function
hypot(3M) ..... Euclidean distance function
matherr(3M) ..... error-handling function
sinh(3M): sinh, cosh, tanh ..... hyperbolic functions
trig(3M): sin, cos, tan, asin, acos,
atan, atan2 ..... trigonometric functions

3N Networking Support Utilities

t_accept(3N) ..... accept a connect request
t_alloc(3N) ..... allocate a library structure
t_bind(3N) ..... bind an address to a transport endpoint
t_close(3N) ..... close a transport endpoint
t_connect(3N) ..... establish a connection with another transport user
t_error(3N) ..... produce error message
t_free(3N) ..... free a library structure
t_getinfo(3N) ..... get protocol-specific service information
t_getstate(3N) ..... get the current state
t_listen(3N) ..... listen for a connect request
t_look(3N) ..... look at the current event on a transport endpoint
t_open(3N) ..... establish a transport endpoint
t_optmgmt(3N) ..... manage options for a transport endpoint
t_rcv(3N) ..... receive data or expedited data sent over a connection
t_rcvconnect(3N) ..... receive the confirmation from a connect request
t_rcvdis(3N) ..... retrieve information from disconnect
t_rcvrel(3N) ..... acknowledge receipt of an orderly release indication

t_rcvudata(3N) ..... receive a data unit
t_rcvuderr(3N) ..... receive a unit data error indication
t_snd(3N) ..... send data or expedited data over a connection
t_snddis(3N) ..... send user-initiated disconnect request
t_sndrel(3N) ..... initiate an orderly release
t_sndudata(3N) ..... send a data unit
t_sync(3N) ..... synchronize transport library
t_unbind(3N) ..... disable a transport endpoint

3PPS Sequent Parallel Programming Library

intro(3PPS) ..... introduction to Parallel Programming Library
brk(3PPS): brk, sbrk ..... change private data segment size
cpus_online(3PPS) ..... returns the number of CPUs on-line
m_fork(3PPS) ..... execute a subprogram in parallel
m_get_myid(3PPS) ..... return process identification
m_get_numprocs(3PPS) ..... return number of child processes
m_kill_procs(3PPS) ..... kill child processes
m_lock(3PPS): m_lock, m_unlock ..... lock, unlock locks
m_next(3PPS) ..... increment global counter
m_park_procs(3PPS): m_park_procs,
m_rele_procs ..... suspend and resume child process execution
m_set_procs(3PPS) ..... set number of child processes
m_single(3PPS): m_single,
m_multi ..... mark single-process code section
m_sync(3PPS) ..... check in at barrier
s_lock(3PPS): s_init_lock, s_lock,
s_clock, s_unlock ..... initialize, lock, unlock locks
s_wait_barrier(3PPS): s_init_barrier,
s_wait_barrier ..... initialize barrier, wait at barrier
shbrk(3PPS): shbrk, shsbrk ..... change shared data segment size
shmalloc(3PPS): shmalloc,
shrealloc, shfree ..... shared memory allocator

3S Standard I/O Library Routines

ctermid(3S) ..... generate filename for terminal
cuserid(3S) ..... get character login name of the user
fclose(3S): fclose, fflush ..... close or flush a stream
ferror(3S): ferror, feof, clearerr,
fileno ..... stream status inquiries
fopen(3S): fopen, freopen, fdopen ..... open a stream
fread(3S): fread, fwrite ..... binary input/output
fseek(3S): fseek, rewind, ftell ..... reposition a file pointer in a stream
getc(3S): getc, getchar, fgetc, getw .... get character or word from a stream
gets(3S): gets, fgets ..... get a string from a stream
popen(3S): popen, pclose ..... initiate pipe to/from a process
printf(3S): printf, fprintf, sprintf ..... print formatted output

putc(3S): putc, putchar, fputc,
putw ..... put character or word on a stream
puts(3S): puts, fputs ..... put a string on a stream
scanf(3S): scanf, fscanf, sscanf ..... convert formatted input
setbuf(3S): setbuf, setvbuf ..... assign buffering to a stream
stdio(3S) ..... standard buffered input/output package
system(3S) ..... issue a shell command
tmpfile(3S) ..... create a temporary file
tmpnam(3S): tmpnam, tempnam ..... create a name for a temporary file
ungetc(3S) ..... push character back into input stream
vprintf(3S): vprintf, vfprintf,
vsprintf ..... print formatted output of a varargs argument list

3SEQ Sequent I/O Library Routines

dis386(3SEQ) ..... disassemble Series 386 instructions
getgeombyname(3SEQ) ..... get disk geometry by name
getmntent(3SEQ): addmntent,
endmntent, getmntent,
hasmntopt, setmntent ..... get file system descriptor file entry
getpseudotty(3SEQ) ..... get pseudo terminals
getscsiinfo(3SEQ): getscsiinfo,
getscsimatch ..... get scsiinfo structure
getusclk(3SEQ): getusclk,
usclk_init ..... get/init microsecond clock
getzdinfo(3SEQ): getzdinfo,
getzdinfobyname ..... get zdinfo structure
initgroups(3SEQ) ..... initialize group access list
nblocks(3SEQ) ..... calculate number of blocks and indirect blocks
offline_all(3SEQ) ..... take all but one active processor off line
read_constab(3SEQ) ..... read constab entries into a structure
valloc(3SEQ) ..... aligned memory allocator

3X Special Routines

assert(3X) ..... verify program assertion
crypt(3X) ..... password and file encryption functions
curses(3X) ..... terminal screen handling and optimization package
directory(3X): opendir, readdir,
telldir, seekdir, rewinddir, closedir .... directory operations
getspent(3X): getspent, getspnam,
setspent, endspent, fgetspent,
lckpwdf, ulckpwdf ..... get shadow password file entry
ldahread(3X) ..... read the archive header of a member of an archive file
ldclose(3X): ldclose, ldaclose ..... close a common object file
ldfhread(3X) ..... read the file header of a common object file
ldgetname(3X) ..... retrieve symbol name for common object file symbol table entry

ldlread(3X): ldlread, ldlinit, ldlitem .. manipulate line number entries of a common object file function
ldlseek(3X): ldlseek, ldnlseek..... seek to line number entries of a section of a common object file
ldohseek(3X) ..... seek to the optional file header of a common object file
ldopen(3X): ldopen, ldaopen ..... open a common object file for reading
ldrseek(3X): ldrseek, ldnrseek ..... seek to relocation entries of a section of a common object file
ldshread(3X): ldshread, ldnshread .... read an indexednamed section header of a common object file
ldsseek(3X): ldsseek, ldnsseek ..... seek to an indexednamed section of a common object file
ldtbindex(3X) ..... compute the index of a symbol table entry of a common object file
ldtbread(3X) ..... read an indexed symbol table entry of a common object file
ldtbseek(3X) ..... seek to the symbol table of a common object file
logname(3X) ..... return login name of user
malloc(3X): malloc, free, realloc, calloc, mallopt, mallinfo ..... fast main memory allocator
ndbm(3X): dbm_open, dbm_close, dbm_fetch, dbm_store, dbm_delete, dbm_firstkey, dbm_nextkey, dbm_error, dbm_clearerr, dmb_blkflush ..... database subroutines
plot(3X) ..... graphics interface subroutines
putspent(3X) ..... write shadow password file entry
regcmp(3X): regcmp, regex ..... compile and execute regular expression
sputl(3X): sputl, sgetl ..... access long integer data in a machine-independent fashion
utadduser(3X): ut_add_user, ut_delete_user ..... add/delete users in utmp file

4 File Formats

intro(4) ..... introduction to file formats
a.out(4) ..... common assembler and link editor output
acct(4) ..... per-process accounting file format
addusr_def(4) ..... add user defaults
alias(4) ..... alias file for FACE
ar(4) ..... see *C and Language Tools Binder*
backup_codes(4) ..... backup cycle code definitions
backup_codes1(4) ..... backup subcode definitions
backup_dev(4) ..... backup devices
backup_info(4) ..... backup set definitions
cftime(4) ..... language specific strings
checklist(4) ..... list of filesystems processed by fsck and ncheck
constab(4) ..... constructed devices table
core(4) ..... format of core image file
cpio(4) ..... format of cpio archive
cronlog(4) ..... cron history file
crontabs(4) ..... clock daemon file
devtab(4) ..... virtual device table

dir(4) ..... format of directories
dirent(4) ..... file system independent directory entry
diskinfo(4) ..... disk device characteristic database
dump(4): dump, dumpdates ..... incremental dump format
env(4): .environ, .pref, .variables ..... FACE files
errorlog(4) ..... internal error log file
errormsg(4) ..... error message text file
filehdr(4) ..... file header for common object files
fs(4): fs, inode ..... format of file system volume
fspec(4) ..... format specification in text files
fstab(4) ..... file-system-table
gettydefs(4) ..... speed and terminal settings used by getty
gps(4): gps ..... graphical primitive string, format of graphical files
group(4) ..... group file
helplog(4) ..... monitor help actions
init.d(4) ..... system level transition files directory
inittab(4) ..... script for the init process
issue(4) ..... issue identification file
ldfcn(4) ..... common object file access routines
limits(4) ..... file header for implementation-specific constants
linenum(4) ..... line number entries in a common object file
loginlog(4) ..... log of failed login attempts
lptypes(4) ..... table of printer types supported by the menu system
mnttab(4) ..... mounted filesystem table
mod_config(4) ..... format of a module configuration file
motd(4) ..... message-of-the-day file
ott(4) ..... files that hold object architecture information
passwd(4) ..... password file
plot(4) ..... graphics interface
product(4) ..... installed products file
profile(4) ..... setting up an environment at login time
proto(4) ..... prototype job file for at
rc(4) ..... reboot and shutdown command scripts
rcsfile(4) ..... format of RCS file
reloc(4) ..... relocation information for a common object file
sccsfile(4) ..... format of SCCS file
scnhdr(4) ..... section header for a common object file
scr_dump(4) ..... format of curses screen image file
shadow(4) ..... restricted password file
shells(4) ..... shell table
shutdown(4) ..... system shutdown information file
spellhist(4) ..... unmatched spelling words file
sulog(4) ..... log of substitute user events
syms(4) ..... common object file symbol table format
sys_config(4) ..... format of a system configuration file
syskeymap(4) ..... identify keys to internal menu system functions
task(4) ..... global security table
term(4) ..... format of compiled term file
termcap(4) ..... terminal capability data base
terminfo(4) ..... terminal capability database

timezone(4) ..... set default system time zone
unistd(4) ..... file header for symbolic constants
utmp(4): utmp, wtmp ..... utmp and wtmp entry formats
uucp(4) ..... networking support files

5 Miscellaneous

intro(5) ..... introduction to miscellany
ascii(5) ..... map of ASCII character set
environ(5) ..... user environment
fcntl(5) ..... file control options
math(5) ..... math functions and constants
prof(5) ..... profile within a function
regexp(5) ..... regular expression compile and match routines
stat(5) ..... data returned by stat system call
term(5) ..... conventional names for terminals
types(5) ..... primitive system data types
values(5) ..... machine-dependent values
varargs(5) ..... handle variable argument list

7 Special Files

intro(7) ..... introduction to special files
clone(7) ..... open any minor device on a STREAMS driver
cmpt(7) ..... compatibility partitioning driver
cn(7) ..... pseudoconsole driver
consmem(7): consmem, sm, ss ..... console memory drivers
console(7) ..... console interface
kmmem(7): kMBmem, kMWmem ..... MULITBUS memory
ld0(7) ..... STREAMS tty line discipline
log(7) ..... interface to STREAMS error logging and event tracing
lp(7) ..... Systech parallel line printer interface
lpmod(7) ..... parallel printer canonical processing module
mbad(7) ..... MULITBUS adapter
mem(7): mem, kmem ..... core memory
mtio(7) ..... UNIX magnetic tape interface
null(7) ..... the null file
pmap(7) ..... map physical addresses into processs virtual address space
sa(7) ..... devices administered by System Administration
sced(7) ..... SCSI/Ethernet/Diagnostics board
sd(7) ..... SCSI disk controller
sp(7) ..... STREAMS pipe device
ssm(7) ..... System Services Module
st(7) ..... Systech MTI-800/1600/1650 terminal multiplexor driver
streamio(7) ..... STREAMS ioctl commands
streampty(7) ..... STREAMS pseudo terminal driver
streamtty(7) ..... Generic STREAMS tty driver interface
termio(7) ..... general terminal interface
tg(7) ..... SCSI 1/2-inch magnetic tape interface

timod(7) ..... Transport Interface cooperating STREAMS module
tirdwr(7) ..... Transport Interface read/write interface STREAMS module
tm(7) ..... SCSI tape controller
ts(7) ..... SCSI tape controller
tty(7) ..... controlling terminal (gentty) interface
usclk(7) ..... 32-bit microsecond counter
vtoc(7) ..... VTOC partitioning driver
wd(7) ..... SSM SCSI disk controller
wp(7) ..... SSM parallel line printer interface
xs(7) ..... Xylogics 781 terminal multiplexor driver
xt(7) ..... Xylogics 472 MULITBUS magnetic tape interface
zd(7) ..... dual-channel disk controller

8 Stand-Alone Utilities

intro(8) ..... introduction to system standalone utilities
ccsformat(8) ..... standalone embedded SCSI disk formatter
boot(8) ..... run a standalone program or bring up the operating system
copy(8) ..... perform a standalone data transfer
dump(8) ..... standalone memory dump
prtvtoc(8) ..... print the VTOC of a standalone disk device
zdformat(8) ..... standalone disk formatter for dual-channel disk controller

Permuted Index

st: Systech MTI-800/1600/	1650 terminal multiplexor driver.....st(7)
hpio: Hewlett-Packard	2645A terminal tape file archiver.....hpio(1)
of DASI 300 and 300s terminals	300, 300s: handle special functions.....300(1)
handle special functions of DASI	300 and 300s terminals.....300(1)
DASI 300 and 300s terminals 300,	300s: handle special functions of.....300(1)
special functions of DASI 300 and	300s terminals.....300(1)
usclk:	32-bit microsecond counter.....usclk(7)
dis386: disassemble Series	386 instructions.....dis386(3SEQ)
DASI 450 terminal	450: handle special functions of the.....450(1)
handle special functions of the DASI	450 terminal.....450(1)
xt: Xylogics	472 MULITBUS magnetic tape interface.....xt(7)
between long integer and base-	64 ASCII string.....a64l(3C)
xs: Xylogics	781 terminal multiplexor driver.....xs(7)
integer and base-64 ASCII string	a64l, l64a: convert between long.....a64l(3C)
	abort: generate an IOT fault.....abort(3C)
	abs: return integer absolute value.....abs(3C)
abs: return integer	absolute value.....abs(3C)
fabs: floor, ceiling, remainder,	absolute value functions.....floor(3M)
t_accept:	accept a connect request.....t_accept(3N)
requests	accept, reject: allow or prevent LP.....accept(1M)
utime: set file	access and modification times.....utime(2)
touch: update	access and modification times of a file.....touch(1)
file	access: determine accessibility of a.....access(2)
initgroups: initialize group	access list.....initgroups(3SEQ)
setgroups: set group	access list.....setgroups(2SEQ)
machine-independent/ sputl, sgetl:	access long integer data in a.....sputl(3X)
ldfcn: common object file	access routines.....ldfcn(4)
endutent, utmpname, getutpid:	access utmp file entry.....getut(3C)
access: determine	accessibility of a file.....access(2)
acct: enable or disable process	accounting.....acct(2)
acctcon1, acctcon2: connect-time	accounting.....acctcon(1M)
acctprc1, acctprc2: process	accounting.....acctprc(1M)
turnacct: shell procedures for	accounting.....acctsh(1M)
runacct: run daily	accounting.....runacct(1M)
/accton, acctwtmp: overview of	accounting and miscellaneous/.....acct(1M)
of accounting and miscellaneous	accounting commands.....acct(1M)
diskusg: generate disk	accounting data by user ID.....diskusg(1M)
acct: per-process	accounting file format.....acct(4)
acctcom: search and print process	accounting files.....acctcom(1)
acctmerg: merge or add total	accounting files.....acctmerg(1M)
command summary from per-process	accounting records.....acctcms(1M)
fwtmp, wtmpfix: manipulate connect	accounting records.....fwtmp(1M)
accounting	acct: enable or disable process.....acct(2)
format	acct: per-process accounting file.....acct(4)
per-process accounting records	acctcms: command summary from.....acctcms(1M)
accounting files	acctcom: search and print process.....acctcom(1)
accounting	acctcon1, acctcon2: connect-time.....acctcon(1M)

acctcon1, acctcon2: connect-time accounting ..... acctcon(1M)
acctwtmp: overview of accounting/ acctdisk, acctdusg, accton, ... acct(1M)
of accounting and/ acctdisk, acctdusg, accton, acctwtmp: overview ..... acct(1M)
accounting files acctmerg: merge or add total ..... acctmerg(1M)
acctdisk, acctdusg, accton, acctwtmp: overview of accounting and/ ..... acct(1M)
accounting acctprc1, acctprc2: process ..... acctprc(1M)
acctprc1, acctprc2: process accounting ..... acctprc(1M)
acctdisk, acctdusg, accton, acctwtmp: overview of accounting and/ ..... acct(1M)
release indication t_rcvrel: acknowledge receipt of an orderly ..... t_rcvrel(3N)
trig: sin, cos, tan, asin, acos, atan, atan2: trigonometric/ ..... trig(3M)
helplog: monitor help actions ..... helplog(4)
sigaction: examine and change signal actions ..... sigaction(2)
killall: kill all active processes ..... killall(1M)
offline_all: take all but one active processor off line ..... offline_all(3SEQ)
mbad: MULITBUS adapter ..... mbad(7)
addcmdopt: add a command option to a menu ..... addcmdopt(1M)
menu addformopt: add a form/SELECT window option to a ..... addformopt(1M)
addmenuopt: add a menu option to a menu ..... addmenuopt(1M)
acctmerg: merge or add total accounting files ..... acctmerg(1M)
addusr.def: add user defaults ..... addusr_def(4)
putenv: change or add value to environment ..... putenv(3C)
menu addcmdopt: add a command option to a ..... addcmdopt(1M)
ut_add_user, ut_delete_user: add/delete users in utmp file ..... utadduser(3X)
option to a menu addformopt: add a form/SELECT window ... addformopt(1M)
menu addmenuopt: add a menu option to a ..... addmenuopt(1M)
hasmntopt, setmntent: get file/ addmntent, endmntent, getmntent, ... getmntent(3SEQ)
map an open file into the processs address space ..... mmap(2SEQ)
munmap: unmap part of the processs address space ..... munmap(2SEQ)
addresses into processs virtual address space ..... pmap(7)
t_bind: bind an address to a transport endpoint ..... t_bind(3N)
pmap: map physical addresses into processs virtual address space ..... pmap(7)
addusr.def: add user defaults ..... addusr_def(4)
files admin: create and administer SCCS ..... admin(1)
print service lpfilter: administer filters used with the LP ..... lpfilter(1M)
print service lpforms: administer forms used with the LP ..... lpforms(1M)
admin: create and administer SCCS files ..... admin(1)
SA: devices administered by System Administration ..... sa(7)
SA: devices administered by System Administration ..... sa(7)
sysadm: menu interface to do system administration ..... sysadm(1)
uadmin: administrative control ..... uadmin(1M)
uadmin: administrative control ..... uadmin(2)
swap: swap administrative interface ..... swap(1M)
/usr/adm/loginlog: log of failed login attempts ..... loginlog(4)
/onproc: run a command with priority aging, PFF, swapping, or processor/ ..... noage(1M)
alarm: set a process alarm clock ..... alarm(2)
alarm: set a process alarm clock ..... alarm(2)
/display application specific alarms and/or the working indicator ..... indicator(1F)
pathalias: alias file for FACE ..... alias(4)
valloc: aligned memory allocator ..... valloc(3SEQ)
t_alloc: allocate a library structure ..... t_alloc(3N)
brk, sbrk: change data segment space allocation ..... brk(2)
free, realloc, calloc: main memory allocator ..... malloc(3C)

mallopt, mallinfo: fast main memory allocator.....malloc(3X)
shrealloc, shfree: shared memory allocator.....shmalloc(3PPS)
valloc: aligned memory allocator.....valloc(3SEQ)
editor output a.out: common assembler and link .....a.out(4)
to maintenance commands and application programs.....intro(1)
the working/ indicator: display application specific alarms and/or.....indicator(1F)
lookup apropos: locate commands by keyword.....apropos(1)
ar: archive file format.....see *C and Lang Tools Binder*
and libraries ar: maintain archives.....see *C and Lang Tools Binder*
bc: arbitrary-precision arithmetic language .....bc(1)
ott: files that hold object architecture information.....ott(4)
cpio: format of cpio archive .....cpio(4)
the archive header of a member of an archive file.....ldahread(3X)
ar: common archive file format .....see *C and Lang Tools Binder*
archive file ldahread: read the archive header of a member of an .....ldahread(3X)
2645A terminal tape file archiver.....hpio(1)
tar: tape file archiver .....tar(1)
cpio: copy file archives in and out.....cpio(1)
varargs: handle variable argument list .....varargs(5)
print formatted output of a varargs argument list .....vprintf(3S)
xargs: construct argument list(s) and execute command.....xargs(1)
getopt: get option letter from argument vector.....getopt(3C)
echo: echo arguments.....echo(1)
expr: evaluate arguments as an expression .....expr(1)
message: puts its arguments on message line .....message(1F)
bc: arbitrary-precision arithmetic language .....bc(1)
asa: interpret ASA carriage control characters.....asa(1)
characters asa: interpret ASA carriage control.....asa(1)
/localtime, gmtime, asctime, cftime, ascftime, mktime, tzset: convert/.....ctime(3C)
ascii: map of ASCII character set .....ascii(5)
ascii: map of ASCII character set.....ascii(5)
between long integer and base-64 ASCII string .....a64l(3C)
ctime, localtime, gmtime, asctime, cftime, ascftime, mktime, tzset:/.....ctime(3C)
trig: sin, cos, tan, asin, acos, atan, atan2: trigonometric/.....trig(3M)
as: common assembler.....see *C and Lang Tools Binder*
a.out: common assembler and link editor output.....a.out(4)
assert: verify program assertion.....assert(3X)
assert: verify program assertion .....assert(3X)
system commands assist: assistance using operating.....assist(1)
/program for generating/modifying ASSIST menus or command forms.....astgen(1)
assist: assistance using operating system commands .....assist(1)
generating/modifying ASSIST menus/ astgen: program for.....astgen(1)
later time at, batch: execute commands at a .....at(1)
trig: sin, cos, tan, asin, acos, atan, atan2: trigonometric functions.....trig(3M)
/sin, cos, tan, asin, acos, atan, atan2: trigonometric functions.....trig(3M)
double-precision number strtod, atof: convert string to .....strtod(3C)
strtol, atol, atoi: convert string to integer.....strtol(3C)
strtol, atol, atoi: convert string to integer.....strtol(3C)
change login password and password attributes.....passwd(1)
control over various process attributes.....proc_ctl(2SEQ)
rcs: change RCS file attributes.....rcs(1)
dumpconf: dump kernel auto-configuration device table.....dumpconf(1M)

language awk: pattern scanning and processing.....awk(1)
backup.codes: backup cycle code definitions.....backup_codes(4)
backup.dev: backup devices.....backup_dev(4)
ckbupscd: check filesystem backup schedule .....ckbupscd(1M)
backup.info: backup set definitions.....backup_info(4)
backup.codes1: backup subcode definitions .....backup_codes1(4)
definitions backup.codes: backup cycle code .....backup_codes(4)
definitions backup.codes1: backup subcode.....backup_codes1(4)
backup.dev: backup devices .....backup_dev(4)
backup.info: backup set definitions.....backup_info(4)
banner: make posters.....banner(1)
m_sync: check in at barrier.....m_sync(3PPS)
initialize barrier, wait at barrier.....s_wait_barrier(3PPS)
/s_wait_barrier: initialize barrier, wait at barrier .....s_wait_barrier(3PPS)
of pathnames basename, dirname: deliver portions.....basename(1)
at, batch: execute commands at a later time.....at(1)
language bc: arbitrary-precision arithmetic .....bc(1)
procedures bcheckrc: system initialization.....bcheckrc(1M)
bdflush: update daemon.....bdflush(1M)
bdflush: update daemon.....bdflush(2SEQ)
bdiff: big diff.....bdiff(1)
bessel: j0, j1, jn, y0, y1, yn: Bessel functions .....bessel(3M)
Bessel functions bessel: j0, j1, jn, y0, y1, yn: .....bessel(3M)
bfs: big file scanner.....bfs(1)
fread, fwrite: binary input/output.....fread(3S)
bp: binary patcher.....bp(1M)
bsearch: binary search a sorted table.....bsearch(3C)
tfind, tdelete, twalk: manage binary search trees.....tsearch(3C)
tmp_affinity: bind a process to a processor.....tmp_affinity(2SEQ)
t_bind: bind an address to a transport endpoint.....t_bind(3N)
sum: print checksum and block count of a file.....sum(1)
prtvtoc: print the VTOC of a block device .....prtvtoc(1M)
sigprocmask: examine and change blocked signals .....sigprocmask(2)
number of blocks and indirect blocks .....nblocks(3SEQ)
nblocks: calculate number of blocks and indirect blocks.....nblocks(3SEQ)
df: report number of free disk blocks and inodes.....df(1M)
consistency check based on the BOM.....bomverify(1M)
consistency check based on the BOM bomverify: perform an installation.....bomverify(1M)
bootflags: display/change system boot flags.....bootflags(1M)
installboot: installs a boot program on a disk.....installboot(1M)
bring up the operating system boot: run a standalone program or.....boot(8)
boot flags bootflags: display/change system .....bootflags(1M)
bp: binary patcher.....bp(1M)
allocation brk, sbrk: change data segment space.....brk(2)
segment size brk, sbrk: change private data.....brk(3PPS)
table bsearch: binary search a sorted .....bsearch(3C)
stdio: standard buffered input/output package .....stdio(3S)
setbuf, setvbuf: assign buffering to a stream.....setbuf(3S)
mailbug: submit a bug report or enhancement request .....mailbug(1)
devbuild, devdestroy: build and destroy virtual devices .....devbuild(1M)
sccstorcs: build RCS file from SCCS file.....sccstorcs(1)
mknod: build special file.....mknod(1M)

config: build system configuration files.....config(1M)
swab: swap bytes.....swab(3C)
cc: C compiler.....see *C and Lang Tools Binder*
cflow: generate C flowgraph.....see *C and Lang Tools Binder*
cpp: the C language preprocessor.....see *C and Lang Tools Binder*
cb: C program beautifier.....see *C and Lang Tools Binder*
lint: a C program checker.....see *C and Lang Tools Binder*
cxref: generate a C program cross-reference.....see *C and Lang Tools Binder*
ctrace: C program debugger.....see *C and Lang Tools Binder*
list: produce C source listing.....see *C and Lang Tools Binder*
cal: print calendar.....cal(1)
indirect blocks nblocks: calculate number of blocks and.....nblocks(3SEQ)
dc: desk calculator.....dc(1)
cal: print calendar.....cal(1)
calendar: reminder service.....calendar(1)
stat: data returned by stat system call.....stat(5)
syscall: indirect system call.....syscall(2)
cu: call another UNIX system.....cu(1C)
malloc, free, realloc, calloc: main memory allocator.....malloc(3C)
malloc, free, realloc, calloc, mallopt, mallinfo: fast main memory/.....malloc(3X)
introduction to Sequent system calls.....intro(2SEQ)
intro: introduction to system calls and error numbers.....intro(2)
lp, cancel: send/ cancel requests to the LP print service.....lp(1)
LP print service lp, cancel: send/cancel requests to the.....lp(1)
lpmod: parallel printer canonical processing module.....lpmod(7)
description into a terminfo/ captoinfo: convert a termcap.....captoinfo(1M)
asa: interpret ASA carriage control characters.....asa(1)
cat: concatenate and print files.....cat(1)
cb: C program beautifier.....see *C and Lang Tools Binder*
gencc: create a front-end to the cc command.....gencc(1M)
cc: C compiler.....see *C and Lang Tools Binder*
disk formatter CCSformat: standalone embedded SCSI.....ccsformat(8)
cd: change working directory.....cd(1)
an SCCS delta cdc: change the delta commentary of.....cdc(1)
remainder, absolute value/ floor, ceil, fmod, fabs: floor, ceiling,.....floor(3M)
floor, ceil, fmod, fabs: floor, ceiling, remainder, absolute value/.....floor(3M)
cflow: generate C flowgraph...see *C and Lang Tools Binder*
ctime, localtime, gmtime, asctime, cftime, ascftime, mktime, tzset:/.....ctime(3C)
cftime: language specific strings.....cftime(4)
sigprocmask: examine and change blocked signals.....sigprocmask(2)
brk, sbrk: change data segment space allocation.....brk(2)
attributes passwd: change login password and password.....passwd(1)
chmod: change mode.....chmod(1)
chmod: change mode of file.....chmod(2)
putenv: change or add value to environment.....putenv(3C)
chown: change owner and group of a file.....chown(2)
chown, chgrp: change owner or group.....chown(1)
nice: change priority of a process.....nice(2)
brk, sbrk: change private data segment size.....brk(3PPS)
rcs: change RCS file attributes.....rcs(1)
chroot: change root directory.....chroot(2)
chroot: change root directory for a command.....chroot(1M)

shbrk, shsbrk: change shared data segment size ..... shbrk(3PPS)
sigaction: examine and change signal actions ..... sigaction(2)
bootflags: display/ change system boot flags ..... bootflags(1M)
shutdown: shut down system, change system state ..... shutdown(1M)
cdc: change the delta commentary of an SCCS delta ..... cdc(1)
newform: change the format of a text file ..... newform(1)
helpadm: change the Help Facility database ..... helpadm(1M)
rename: change the name of a file ..... rename(2)
delta: make a delta (change) to an SCCS file ..... delta(1)
parameters vm_ctl: examine and change virtual memory tuning ..... vm_ctl(2SEQ)
cd: change working directory ..... cd(1)
chdir: change working directory ..... chdir(2)
pipe: create an interprocess channel ..... pipe(2)
zd: dual- channel disk controller ..... zd(7)
standalone disk formatter for dual- channel disk controller ..... zdformat(8)
strstr: find first occurrence of character ..... strstr(3C)
ungetc: push character back into input stream ..... ungetc(3S)
conversion tables chrtbl: generate character classification and ..... chrtbl(1M)
_tolower, _toupper, setchrclass: character handling ..... ctype(3C)
cuserid: get character login name of the user ..... cuserid(3S)
getc, getchar, fgetc, getw: get character or word from a stream ..... getc(3S)
putc, putchar, fputc, putw: put character or word on a stream ..... putc(3S)
ascii: map of ASCII character set ..... ascii(5)
fgrep: search a file for a character string ..... fgrep(1)
diskinfo: disk device characteristic database ..... diskinfo(4)
asa: interpret ASA carriage control characters ..... asa(1)
_tolower, toascii: translate characters ..... conv(3C)
tr: translate characters ..... tr(1)
lastlogin, monacct, nulladm/ chargefee, ckpacct, dodisk, ... acctsh(1M)
chdir: change working directory ..... chdir(2)
filesystem directory consistency check ..... dcheck(1M)
filesystem storage consistency check ..... icheck(1M)
checkfsys: check a filesystem ..... checkfsys(1M)
fsck: filesystem consistency check and interactive repair ..... fsck(1M)
perform an installation consistency check based on the BOM ..... bomverify(1M)
checkeq: check documents for correct eqn syntax ..... checkeq(1)
ckbupscd: check filesystem backup schedule ..... ckbupscd(1M)
m_sync: check in at barrier ..... m_sync(3PPS)
ci: check in RCS revisions ..... ci(1)
co: check out RCS revisions ..... co(1)
permissions file uucheck: check the uucp directories and ..... uucheck(1M)
eqn syntax. checkeq: check documents for correct ..... checkeq(1)
checkfsys: check a filesystem ..... checkfsys(1M)
processed by fsck and ncheck checklist: list of filesystems ..... checklist(4)
sum: print checksum and block count of a file ..... sum(1)
chown, chgrp: change owner or group ..... chown(1)
/m_rele_procs: suspend and resume child process execution ..... m_park_procs(3PPS)
times: get process and child process times ..... times(2)
wait, waitpid: wait for a child process to stop or terminate ..... wait(2)
m_get_numprocs: return number of child processes ..... m_get_numprocs(3PPS)
m_kill_procs: kill child processes ..... m_kill_procs(3PPS)
m_set_procs: set number of child processes ..... m_set_procs(3PPS)

chmod: change mode ..... chmod(1)
chmod: change mode of file ..... chmod(2)
file chown: change owner and group of a ..... chown(2)
chown, chgrp: change owner or group ..... chown(1)
chroot: change root directory ..... chroot(2)
command chroot: change root directory for a ..... chroot(1M)
classification and conversion/ chrtbl: generate character ..... chrtbl(1M)
ci: check in RCS revisions ..... ci(1)
schedule ckbupscd: check filesystem backup ..... ckbupscd(1M)
nulladm, prctmp,/ chargefee, ckpacct, dodisk, lastlogin, monacct, ... acctsh(1M)
chrtbl: generate character classification and conversion tables ..... chrtbl(1M)
clri: clear inode ..... clri(1M)
ferror, feof, clearerr, fileno: stream status inquiries ..... ferror(3S)
a shell (command interpreter) with C-like syntax ..... csh(1)
alarm: set a process alarm clock ..... alarm(2)
usclk_init: get/init microsecond clock ..... getusclk(3SEQ)
cron: clock daemon ..... cron(1M)
crontabs: clock daemon file ..... crontabs(4)
clock: report CPU time used ..... clock(3C)
STREAMS driver clone: open any minor device on a ..... clone(7)
ldclose, ldaclose: close a common object file ..... ldclose(3X)
close: close a file descriptor ..... close(2)
t_close: close a transport endpoint ..... t_close(3N)
close: close a file descriptor ..... close(2)
fclose, fflush: close or flush a stream ..... fclose(3S)
/telldir, seekdir, rewinddir, closedir: directory operations ..... directory(3X)
clri: clear inode ..... clri(1M)
cmp: compare two files ..... cmp(1)
driver cmpt: compatibility partitioning ..... cmpt(7)
cn: pseudoconsole driver ..... cn(7)
co: check out RCS revisions ..... co(1)
communicate to a/ cocreate, cosend, cocheck, coreceive, codestroy: ..... coproc(1F)
coreceive, codestroy: communicate/ cocreate, cosend, cocheck, ... coproc(1F)
backup.codes: backup cycle code definitions ..... backup_codes(4)
m_multi: mark single-process code section ..... m_single(3PPS)
/cosend, cocheck, coreceive, codestroy: communicate to a process ..... coproc(1F)
col: filter reverse line-feeds ..... col(1)
to form error log dmesg: collect system diagnostic messages ..... dmesg(1M)
setcolor: redefine or create a color ..... setcolor(1F)
comb: combine SCCS deltas ..... comb(1)
comb: combine SCCS deltas ..... comb(1)
to two sorted files comm: select or reject lines common ..... comm(1)
mcs: manipulate the object file comment section ..... mcs(1)
cdc: change the delta commentary of an SCCS delta ..... cdc(1)
a.out: common assembler and link editor output ..... a.out(4)
ldclose, ldaclose: close a common object file ..... ldclose(3X)
ldfhread: read the file header of a common object file ..... ldfhread(3X)
number entries of a section of a common object file ..... ldlseek(3X)
to the optional file header of a common object file ..... ldohseek(3X)
relocation entries of a section of a common object file ..... ldrseek(3X)
indexednamed section header of a common object file ..... ldshread(3X)
to an indexednamed section of a common object file ..... ldsseek(3X)

index of a symbol table entry of a common object file ..... ldtbindex(3X)
an indexed symbol table entry of a common object file ..... ldtbread(3X)
seek to the symbol table of a common object file ..... ldtbseek(3X)
linenum: line number entries in a common object file ..... linenum(4)
reloc: relocation information for a common object file ..... reloc(4)
scnhdr: section header for a common object file ..... scnhdr(4)
ldfcn: common object file access routines ..... ldfcn(4)
ldopen, ldaopen: open a common object file for reading ..... ldopen(3X)
/manipulate line number entries of a common object file function ..... ldlread(3X)
ldgetname: retrieve symbol name for common object file symbol table/ ..... ldgetname(3X)
syms: common object file symbol table format ..... syms(4)
filehdr: file header for common object files ..... filehdr(4)
symbols glossary: definitions of common operating system terms and ..... glossary(1)
comm: select or reject lines common to two sorted files ..... comm(1)
cocheck, coreceive, codestroy: communicate to a process ..... coproc(1F)
ipcs: report inter-process communication facilities status ..... ipcs(1)
stdipc: ftok: standard interprocess communication package ..... stdipc(3C)
tfipc: interprocess communication reporter ..... tfipc(1M)
descriptions infocmp: compare or print out terminfo ..... infocmp(1M)
rcsdiff: compare RCS revisions ..... rcsdiff(1)
cmp: compare two files ..... cmp(1)
sccsdiff: compare two versions of an SCCS file ..... sccsdiff(1)
diff: differential file comparison ..... diff(1)
diff3: 3-way differential file comparison ..... diff3(1)
dircmp: directory comparison ..... dircmp(1)
cmpt: compatibility partitioning driver ..... cmpt(7)
regcmp: regular expression compile ..... regcmp(1)
regcmp, regex: compile and execute regular expression ..... regcmp(3X)
regexp: regular expression compile and match routines ..... regexp(5)
term: format of compiled term file ..... term(4)
cc: C compiler ..... see *C and Lang Tools Binder*
ec: forms compiler ..... ec(1M)
tic: terminfo compiler ..... tic(1M)
yacc: yet another compiler- compiler ..... yacc(1)
yacc: yet another compiler-compiler ..... yacc(1)
erf, erfc: error function and complementary error function ..... erf(3M)
cprs: compress an object file ..... see *C and Lang Tools Binder*
compress, uncompress, zcat: compress and expand data ..... compress(1)
pack, pcat, unpack: compress and expand files ..... pack(1)
and expand data compress, uncompress, zcat: compress ..... compress(1)
entry of a common object/ ldtbindex: compute the index of a symbol table ..... ldtbindex(3X)
cat: concatenate and print files ..... cat(1)
test: condition evaluation command ..... test(1)
files config: build system configuration ..... config(1M)
pathconf, fpathconf: get configurable pathname variables ..... pathconf(2)
sysconf: get configurable system variables ..... sysconf(2)
dumpconf: dump kernel auto- configuration device table ..... dumpconf(1M)
mod_config: format of a module configuration file ..... mod_config(4)
sys_config: format of a system configuration file ..... sys_config(4)
config: build system configuration files ..... config(1M)
showcfg: show the configuration of the machine ..... showcfg(1M)
lpadmin: configure the LP print service ..... lpadmin(1M)

t_rcvconnect: receive the — confirmation from a connect request ..... t_rcvconnect(3N)
fwtmp, wtmpfix: manipulate — connect accounting records ..... fwtmp(1M)
t_accept: accept a — connect request ..... t_accept(3N)
t_listen: listen for a — connect request ..... t_listen(3N)
receive the confirmation from a — connect request ..... t_rcvconnect(3N)
establish an out-going terminal line — connection ..... dial(3C)
data or expedited data sent over a — connection ..... t_rcv(3N)
send data or expedited data over a — connection ..... t_snd(3N)
t_connect: establish a — connection with another transport user ..... t_connect(3N)
acctcon1, acctcon2: — connect-time accounting ..... acctcon(1M)
dcheck: filesystem directory — consistency check ..... dcheck(1M)
icheck: filesystem storage — consistency check ..... icheck(1M)
fsck: filesystem — consistency check and interactive repair ..... fsck(1M)
bomverify: perform an installation — consistency check based on the BOM ..... bomverify(1M)
quotacheck: filesystem quota — consistency checker ..... quotacheck(1M)
drivers — consmem, sm, ss: console memory ..... consmem(7)
— console: console interface ..... console(7)
console: — console interface ..... console(7)
consmem, sm, ss: — console memory drivers ..... consmem(7)
— constab: constructed devices table ..... constab(4)
read_constab: read — constab entries into a structure ..... read_constab(3SEQ)
header for implementation-specific — constants ..... limits(4)
math: math functions and — constants ..... math(5)
unistd: file header for symbolic — constants ..... unistd(4)
constab: — constructed devices table ..... constab(4)
deroff: remove nroff, tbl, and eqn — constructs ..... deroff(1)
Uutry: try to — contact remote system with debugging on ..... uutry(1M)
ls: list — contents of directory ..... ls(1)
sigcontext: get signal — context ..... sigcontext(2SEQ)
set and/or get signal stack — context ..... sigstack(2SEQ)
csplit: — context split ..... csplit(1)
sd: SCSI disk — controller ..... sd(7)
tm: SCSI tape — controller ..... tm(7)
ts: SCSI tape — controller ..... ts(7)
wd: SSM SCSI disk — controller ..... wd(7)
zd: dual-channel disk — controller ..... zd(7)
disk formatter for dual-channel disk — controller ..... zdformat(8)
synchronize a co-process with the — controlling FMLI object ..... vsig(1F)
virtually hangup the specified — controlling terminal ..... fvhangup(2SEQ)
tty: — controlling terminal (gentty) interface ..... tty(7)
_tolower, toascii: translate/ — conv: toupper, tolower, _toupper, ... conv(3C)
units: — conversion program ..... units(1)
character classification and — conversion tables ..... chrtbl(1M)
terminfo description captoinfo: — convert a termcap description into a ..... captoinfo(1M)
dd: — convert and copy a file ..... dd(1M)
long integers l3tol, ltol3: — convert between 3-byte integers and ..... l3tol(3C)
base-64 ASCII string a64l, l64a: — convert between long integer and ..... a64l(3C)
dev_to_ttyname, ttyname_to_dev: — convert major/minor numbers ..... dev_to_ttyname(3C)
/cftime, ascftime, mktime, tzset: — convert date and time to string ..... ctime(3C)
ecvt, fcvt, gcvt: — convert floating-point number to string ..... ecvt(3C)
scanf, fscanf, sscanf: — convert formatted input ..... scanf(3S)
strtod, atof: — convert string to double-precision number ..... strtod(3C)

strtol, atol, atoi: convert string to integer ..... strtol(3C)
password file scheme pwunconv: converts from a two to a one ..... pwunconv(1M)
vsig: synchronize a co-process with the controlling FMLI object ..... vsig(1F)
UNIX-to-UNIX system copy ..... uucp(1C)
system file copy ..... uuto(1C)
dd: convert and copy a file ..... dd(1M)
cpset: copy file and set mode, user, and group IDs ..... cpset(1)
cpio: copy file archives in and out ..... cpio(1)
cp, ln, mv: copy, link or move files ..... cp(1)
volcopy: make literal copy of filesystem ..... volcopy(1M)
transfer copy: perform a standalone data ..... copy(8)
savecore: save a core dump of the operating system ..... savecore(1M)
core: format of core image file ..... core(4)
core: format of core image file ..... core(4)
mem, kmem: core memory ..... mem(7)
and unset environment variables in core or in files ..... set(1F)
cocreate, cosend, cocheck, coreceive, codestroy: communicate to a/ ..... coproc(1F)
checkeq: check documents for correct eqn syntax ..... checkeq(1)
trigonometric functions trig: sin, cos, tan, asin, acos, atan, atan2: ..... trig(3M)
codestroy: communicate to/ cocreate, cosend, cocheck, coreceive, ... coproc(1F)
sinh, cosh, tanh: hyperbolic functions ..... sinh(3M)
wc: word count ..... wc(1)
sum: print checksum and block count of a file ..... sum(1)
m_next: increment global counter ..... m_next(3PPS)
usclk: 32-bit microsecond counter ..... usclk(7)
cp, ln, mv: copy, link or move files ..... cp(1)
cpio: format of cpio archive ..... cpio(4)
cpio: copy file archives in and out ..... cpio(1)
cpio: format of cpio archive ..... cpio(4)
cpp: C preprocessor ..... see *C and Lang Tools Binder*
cprs: compress object file ..... see *C and Lang Tools Binder*
and group IDs cpset: copy file and set mode, user, ... cpset(1)
clock: report CPU time used ..... clock(3C)
cpus_online: returns the number of CPUs on-line ..... cpus_online(3PPS)
CPUs on-line cpus_online: returns the number of ..... cpus_online(3PPS)
crash: examine system images ..... crash(1M)
an existing one creat: create a new file or rewrite ..... creat(2)
setcolor: redefine or create a color ..... setcolor(1F)
gencc: create a front-end to the cc command ..... gencc(1M)
tmpnam, tempnam: create a name for a temporary file ..... tmpnam(3S)
creat: create a new file or rewrite an existing one ..... creat(2)
fork: create a new process ..... fork(2)
ctags: create a tags file ..... ctags(1)
tmpfile: create a temporary file ..... tmpfile(3S)
pipe: create an interprocess channel ..... pipe(2)
admin: create and administer SCCS files ..... admin(1)
tasktbl: create security table in shared memory ..... tasktbl(1M)
setsid: create session and set process group ID ..... setsid(2)
pathconv: search Interpreter criteria for filename ..... pathconv(1F)
cron: clock daemon ..... cron(1M)
cronlog: cron history file ..... cronlog(4)
cronlog: cron history file ..... cronlog(4)

crontab: user — crontab file ..... crontab(1)
crontab: user crontab file ..... crontab(1)
crontabs: clock daemon file ..... crontabs(4)
cxref: generate C program — cross-reference ..... see *C and Lang Tools Binder*
pg: file perusal filter for — CRTs ..... pg(1)
crypt: encode/decode ..... crypt(1)
functions — crypt: password and file encryption ..... crypt(3X)
hashing encryption — crypt, setkey, encrypt: generate ..... crypt(3C)
with C-like syntax — csh: a shell (command interpreter) ..... csh(1)
csplit: context split ..... csplit(1)
ct: spawn getty to a remote terminal ..... ct(1C)
ctags: create a tags file ..... ctags(1)
terminal — ctermid: generate filename for ..... ctermid(3S)
cftime, ascftime, mktime, tzset:/ — ctime, localtime, gmtime, asctime, ... ctime(3C)
ctrace: C prog debugger ..... see *C and Lang Tools Binder*
cu: call another UNIX system ..... cu(1C)
scr_dump: format of — curses screen image file ..... scr_dump(4)
optimization package — curses: terminal screen handling and ..... curses(3X)
the user — cuserid: get character login name of ..... cuserid(3S)
line of a file — cut: cut out selected fields of each ..... cut(1)
cut: — cut out selected fields of each line of a file ..... cut(1)
cross-reference — cxref: generate C program ..... see *C and Lang Tools Binder*
backup.codes: backup — cycle code definitions ..... backup_codes(4)
bdflush: update — daemon ..... bdflush(1M)
bdflush: update — daemon ..... bdflush(2SEQ)
cron: clock — daemon ..... cron(1M)
strerr: STREAMS error logger — daemon ..... strerr(1M)
crontabs: clock — daemon file ..... crontabs(4)
creation pwdhd: — daemon to oversee password database ..... pwdhd(1M)
runacct: run — daily accounting ..... runacct(1M)
300s: handle special functions of — DASI 300 and 300s terminals ..... 300(1)
450: handle special functions of the — DASI 450 terminal ..... 450(1)
zcat: compress and expand — data ..... compress(1)
prof: display profile — data ..... prof(1)
time a command; report process — data and system activity ..... timex(1)
termcap: terminal capability — data base ..... termcap(4)
diskusg: generate disk accounting — data by user ID ..... diskusg(1M)
t_rcvuderr: receive a unit — data error indication ..... t_rcvuderr(3N)
sputl, sgetl: access long integer — data in a machine-independent/ ..... sputl(3X)
plock: lock process, text, or — data in memory ..... plock(2)
t_snd: send — data or expedited data over a connection ..... t_snd(3N)
t_rcv: receive — data or expedited data sent over a connection ..... t_rcv(3N)
t_snd: send data or expedited — data over a connection ..... t_snd(3N)
stat: — data returned by stat system call ..... stat(5)
brk, sbrk: change private — data segment size ..... brk(3PPS)
shbrk, shsbrk: change shared — data segment size ..... shbrk(3PPS)
brk, sbrk: change — data segment space allocation ..... brk(2)
t_rcv: receive data or expedited — data sent over a connection ..... t_rcv(3N)
copy: perform a standalone — data transfer ..... copy(8)
types: primitive system — data types ..... types(5)
t_rcvudata: receive a — data unit ..... t_rcvudata(3N)
t_sndudata: send a — data unit ..... t_sndudata(3N)

diskinfo: disk device characteristic database.....diskinfo(4)
helpadm: change the Help Facility database.....helpadm(1M)
terminfo: terminal capability database.....terminfo(4)
a terminal or query terminfo database.....tput(1)
pwdhd: daemon to oversee password database creation.....pwdhd(1M)
mkpwdbm: make a database for fast password lookup.....mkpwdbm(1M)
join: relational database operator.....join(1)
dbm_clearerr, dmb_blkflush: database subroutines.....ndbm(3X)
makewhatis: generate man page databases.....makealiases(1M)
date: print and set the date.....date(1)
ascftime, mktime, tzset: convert date and time to string.....ctime(3C)
date: print and set the date.....date(1)
/dbm_nextkey, dbm_error, dbm_clearerr, dmb_blkflush: database/.....ndbm(3X)
dbm_delete,/ dbm_open, dbm_close, dbm_fetch, dbm_store,.....ndbm(3X)
dbm_close, dbm_fetch, dbm_store, /dbm_firstkey, dbm_nextkey,.....ndbm(3X)
/dbm_firstkey, dbm_nextkey, dbm_error,/.....ndbm(3X)
dbm_firstkey,/dbm_open, dbm_close, dbm_fetch, dbm_store, dbm_delete,.....ndbm(3X)
/dbm_fetch, dbm_store, dbm_delete, dbm_firstkey, dbm_nextkey,/.....ndbm(3X)
dbm_store, dbm_delete, dbm_firstkey, dbm_nextkey, dbm_error,/.....ndbm(3X)
dbm_store, dbm_delete,/ dbm_open, dbm_close, dbm_fetch, ...ndbm(3X)
dbm_open, dbm_close, dbm_fetch, pdbx, dbm_store, dbm_delete, dbm_firstkey,/.....ndbm(3X)
pdbx, dbx: parallel debugger.....see *C and Lang Tools Binder*
dc: desk calculator.....dc(1)
enable/disable patrol seeks on DCC disks.....sweepon(1M)
consistency check dcheck: filesystem directory.....dcheck(1M)
dd: convert and copy a file.....dd(1M)
ctrace: C program debugger.....see *C and Lang Tools Binder*
pdbx, dbx: parallel debugger.....see *C and Lang Tools Binder*
try to contact remote system with debugging on.....uutry(1M)
crypt: encode/ decode.....crypt(1)
timezone: set default system time zone.....timezone(4)
reset the current form field to its default values.....reset(1F)
addusr.def: add user defaults.....addusr_def(4)
format disks or perform other defect management functions.....format(1M)
gtty: set and get terminal state (defunct).....stty(2)
ut_add_user, ut_delete_user: add/ delete users in utmp file.....utadduser(3X)
the delta commentary of an SCCS delta.....cdc(1)
delta: make a delta (change) to an SCCS file.....delta(1)
cdc: change the delta commentary of an SCCS delta.....cdc(1)
rmdel: remove a delta from an SCCS file.....rmdel(1)
SCCS file delta: make a delta (change) to an.....delta(1)
comb: combine SCCS deltas.....comb(1)
constructs deroff: remove nroff, tbl, and eqn.....deroff(1)
whatis: describe what a command is.....whatis(1)
close: close a file descriptor.....close(2)
dup: duplicate an open file descriptor.....dup(2)
dup2: duplicate an open file descriptor.....dup2(2)
setmntent: get file system descriptor file entry.....getmntent(3SEQ)
getdtablemax: get maximum in-use descriptor table index.....getdtablemax(2SEQ)
getdtablesize: get descriptor table size.....getdtablesize(2SEQ)
setdtablesize: set descriptor table size.....setdtablesize(2SEQ)
dc: desk calculator.....dc(1)

devbuild, devdestroy: build and destroy virtual devices ... devbuild(1M)
destroy virtual devices ... devbuild(1M)
devbuild, devdestroy: build and ... devbuild(1M)
virtual devices devbuild, devdestroy: build and destroy ... devbuild(1M)
ioctl: control device ... ioctl(2)
prtvtoc: print the VTOC of a block device ... prtvtoc(1M)
print the VTOC of a standalone disk device ... prtvtoc(8)
sp: STREAMS pipe device ... sp(7)
or the terminal associated with a device ... ttydevname(1)
the terminal/ ttydevname: get the device associated with a terminal or ... ttydevname(1)
diskinfo: disk device characteristic database ... diskinfo(4)
devnm: device name ... devnm(1M)
clone: open any minor device on a STREAMS driver ... clone(7)
devtab: virtual device table ... devtab(4)
dump kernel auto-configuration device table ... dumpconf(1M)
backup.dev: backup devices ... backup_dev(4)
build and destroy virtual devices ... devbuild(1M)
of virtual-to-physical mapping devices ... pmap(1M)
SA: devices administered by System Administration ... sa(7)
constab: constructed devices table ... constab(4)
devnm: device name ... devnm(1M)
devtab: virtual device table ... devtab(4)
convert between major/minor numbers/ dev_to_ttyname, ttyname_to_dev: ... dev_to_ttyname(3C)
blocks and inodes df: report number of free disk ... df(1M)
dmesg: collect system diagnostic messages to form error log ... dmesg(1M)
SCED: SCSI/Ethernet/ Diagnostics board ... sced(7)
terminal line connection dial: establish an out-going ... dial(3C)
bdiff: big diff ... bdiff(1)
diff: differential file comparison ... diff(1)
comparison diff3: 3-way differential file ... diff3(1)
sdiff: side-by-side difference program ... sdiff(1)
diff: differential file comparison ... diff(1)
diff3: 3-way differential file comparison ... diff3(1)
dir: format of directories ... dir(4)
dircmp: directory comparison ... dircmp(1)
dir: format of directories ... dir(4)
unlink: link and unlink files and directories ... link(1M)
mkdir: make directories ... mkdir(1)
rm, rmdir: remove files or directories ... rm(1)
uucheck: check the uucp directories and permissions file ... uucheck(1M)
cd: change working directory ... cd(1)
chdir: change working directory ... chdir(2)
chroot: change root directory ... chroot(2)
get pathname of current working directory ... getcwd(3C)
system level transition files directory ... init.d(4)
ls: list contents of directory ... ls(1)
mkdir: make a directory ... mkdir(2)
mvdir: move a directory ... mvdir(1M)
rmdir: remove a directory ... rmdir(2)
uucleanup: uucp spool directory clean-up ... uucleanup(1M)
dircmp: directory comparison ... dircmp(1)
dcheck: filesystem directory consistency check ... dcheck(1M)
system independent/ getdents: read directory entries and put in a file ... getdents(2)

dirent: file system independent — directory entry ..... dirent(4)
unlink: remove — directory entry ..... unlink(2)
chroot: change root — directory for a command ..... chroot(1M)
pwd: working — directory name ..... pwd(1)
telldir, seekdir, rewinddir/ — directory: opendir, readdir, ... directory(3X)
seekdir, rewinddir, closedir: — directory operations ..... directory(3X)
mknod: make a — directory, or a special or ordinary file ..... mknod(2)
directory entry — dirent: file system independent ..... dirent(4)
basename, — dirname: deliver portions of pathnames ..... basename(1)
— dis: code disassembler ..... see *C and Lang Tools Binder*
instructions — dis386: disassemble Series 386 ..... dis386(3SEQ)
t_unbind: — disable a transport endpoint ..... t_unbind(3N)
enable, — disable: enable/disable LP printers ..... enable(1)
enable, disable: enable/ — disable LP printers ..... enable(1)
sweepon, sweepoff: enable/ — disable patrol seeks on DCC disks ..... sweepon(1M)
acct: enable or — disable process accounting ..... acct(2)
swapping, or processor migration — disabled ..... noage(1M)
dis386: — disassemble Series 386 instructions ..... dis386(3SEQ)
dis: object code — disassembler ..... see *C and Lang Tools Binder*
t_rcvdis: retrieve information from — disconnect ..... t_rcvdis(3N)
t_snddis: send user-initiated — disconnect request ..... t_snddis(3N)
installs a boot program on a — disk ..... installboot(1M)
diskusg: generate — disk accounting data by user ID ..... diskusg(1M)
df: report number of free — disk blocks and inodes ..... df(1M)
sd: SCSI — disk controller ..... sd(7)
wd: SSM SCSI — disk controller ..... wd(7)
zd: dual-channel — disk controller ..... zd(7)
disk formatter for dual-channel — disk controller ..... zdformat(8)
print the VTOC of a standalone — disk device ..... prtvtoc(8)
diskinfo: — disk device characteristic database ..... diskinfo(4)
CCSformat: standalone embedded SCSI — disk formatter ..... ccsformat(8)
controller zdformat: standalone — disk formatter for dual-channel disk ..... zdformat(8)
getgeombyname: get — disk geometry by name ..... getgeombyname(3SEQ)
quotactl: manipulate — disk quotas ..... quotactl(2SEQ)
du: summarize — disk usage ..... du(1M)
quota: display — disk usage and limits ..... quota(1)
database — diskinfo: disk device characteristic ..... diskinfo(4)
mkvtoc: populate VTOC on — disks ..... mkvtoc(1M)
enable/disable patrol seeks on DCC — disks ..... sweepon(1M)
mgmt functions format: format — disks or perform other defect ..... format(1M)
data by user ID — diskusg: generate disk accounting ..... diskusg(1M)
and/or the working/ indicator: — display application specific alarms ..... indicator(1F)
quota: — display disk usage and limits ..... quota(1)
vi: screen-oriented (visual) — display editor based on ex ..... vi(1)
man: — display on-line man pages ..... man(1)
virtual-to-physical mapping/ pmap: — display or alter parameters of ..... pmap(1M)
prof: — display profile data ..... prof(1)
bootflags: — display/change system boot flags ..... bootflags(1M)
hypot: Euclidean — distance function ..... hypot(3M)
— dltsectbl: refresh security table ..... dltsectbl(1M)
/dbm_error, dbm_clearerr, — dmb_blkflush: database subroutines ..... ndbm(3X)
messages to form error log — dmesg: collect system diagnostic ..... dmesg(1M)

checkeq: check documents for correct eqn syntax.....checkeq(1)
prctmp,/ chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm,.....acctsh(1M)
strtod, atof: convert string to double-precision number.....strtod(3C)
mrand48, jrand48, srand48, seed48,/ drand48, erand48, lrand48, nrand48,.....drand48(3C)
open any minor device on a STREAMS driver.....clone(7)
cmpt: compatibility partitioning driver.....cmpt(7)
cn: pseudoconsole driver.....cn(7)
terminal multiplexor driver.....st(7)
streampty: STREAMS pseudo terminal driver.....streampty(7)
vtoc: VTOC partitioning driver.....vtoc(7)
Xylogics 781 terminal multiplexor driver.....xs(7)
streamtty: Generic STREAMS tty driver interface.....streamtty(7)
consmem, sm, ss: console memory drivers.....consmem(7)
du: summarize disk usage.....du(1M)
zd: dual-channel disk controller.....zd(7)
/standalone disk formatter for dual-channel disk controller.....zdformat(8)
dump: incremental filesystem dump.....dump(1M)
dump: standalone memory dump.....dump(8)
od: octal dump.....od(1)
of an object file dump: dump parts.....see *C and Lang Tools Binder*
format dump, dumpdates: incremental dump.....dump(4)
dumpfs: dump filesystem information.....dumpfs(1M)
dump, dumpdates: incremental dump format.....dump(4)
dump: incremental filesystem dump.....dump(1M)
device table dumpconf: dump kernel auto-configuration.....dumpconf(1M)
savecore: save a core dump of the operating system.....savecore(1M)
dump: standalone memory dump.....dump(8)
auto-configuration device table dumpconf: dump kernel.....dumpconf(1M)
dump, dumpdates: incremental dump format.....dump(4)
dumpfs: dump filesystem information.....dumpfs(1M)
descriptor dup: duplicate an open file.....dup(2)
descriptor dup2: duplicate an open file.....dup2(2)
dup: duplicate an open file descriptor.....dup(2)
dup2: duplicate an open file descriptor.....dup2(2)
ec: forms compiler.....ec(1M)
echo: echo arguments.....echo(1)
echo: echo arguments.....echo(1)
echo: put string on virtual output.....echo(1F)
floating-point number to string ecvt, fcvt, gcvt: convert.....ecvt(3C)
ed, red: text editor.....ed(1)
end, etext, edata: last locations in program.....end(3C)
command edc: run an externally described.....edc(1M)
casual users) edit: text editor (variant of ex for.....edit(1)
edquota: edit user quotas.....edquota(1M)
sact: print current SCCS file editing activity.....sact(1)
ed, red: text editor.....ed(1)
ex: text editor.....ex(1)
sed: stream editor.....sed(1)
vi: screen-oriented (visual) display editor based on ex.....vi(1)
ld: link editor for object files.....see *C and Lang Tools Binder*
a.out: common assembler and link editor output.....a.out(4)
edit: text editor (variant of ex for casual users).....edit(1)

edquota: edit user quotas ..... edquota(1M)
effective user, real group, and effective group IDs ..... getuid(2)
/getgid, getegid: get real user, effective user, real group, and/ ..... getuid(2)
efl: Extended Fortran Language ..... efl(1)
fsplit: split f77, ratfor, or efl files ..... fsplit(1)
using full regular expressions egrep: search a file for a pattern ..... egrep(1)
CCSformat: standalone embedded SCSI disk formatter ..... ccsformat(8)
printers enable, disable: enable/disable LP ..... enable(1)
acct: enable or disable process accounting ..... acct(2)
enable, disable: enable/disable LP printers ..... enable(1)
sweepon, sweepoff: enable/disable patrol seeks on DCC disks ..... sweepon(1M)
crypt: encode/decode ..... crypt(1)
crypt, setkey, encrypt: generate hashing encryption ..... crypt(3C)
setkey, encrypt: generate hashing encryption ..... crypt(3C)
crypt: password and file encryption functions ..... crypt(3X)
makekey: generate encryption key ..... makekey(1)
program end, etext, edata: last locations in ..... end(3C)
gencc: create a front- end to the cc command ..... gencc(1M)
/getgrgid, getgrnam, setgrent, endgrent, fgetgrent: get group file entry ..... getgrent(3C)
setmntent: get file/ addmntent, endmntent, getmntent, hasmntopt, ... getmntent(3SEQ)
bind an address to a transport endpoint ..... t_bind(3N)
t_close: close a transport endpoint ..... t_close(3N)
at the current event on a transport endpoint ..... t_look(3N)
t_open: establish a transport endpoint ..... t_open(3N)
manage options for a transport endpoint ..... t_optmgmt(3N)
t_unbind: disable a transport endpoint ..... t_unbind(3N)
/getpwuid, getpwnam, setpwent, endpwent, fgetpwent: get password file/ ..... getpwent(3C)
getspent, getspnam, setspent, endspent, fgetspent, lckpwdf,/ ..... getspent(3X)
/getutline, pututline, setutent, endutent, utmpname, getutpid: access/ ..... getut(3C)
mailbug: submit a bug report or enhancement request ..... mailbug(1)
execution env: set environment for command ..... env(1)
files .environ, .pref, .variables: FACE ..... env(4)
environ: user environment ..... environ(5)
environ: user environment ..... environ(5)
putenv: change or add value to environment ..... putenv(3C)
commands performed for multi-user environment ..... rc2(1M)
profile: setting up an environment at login time ..... profile(4)
fpsetsticky: IEEE floating-point environment control ..... fpgetround(3C)
env: set environment for command execution ..... env(1)
getenv: return value for environment name ..... getenv(3C)
set, unset: set and unset environment variables in core or in files ..... set(1F)
deroff: remove nroff, tbl, and eqn constructs ..... deroff(1)
checkeq: check documents for correct eqn syntax ..... checkeq(1)
jrand48, srand48, seed48,/ drand48, erand48, lrand48, nrand48, mrand48, ... drand48(3C)
complementary error function erf, erfc: error function and ..... erf(3M)
complementary error function erf, erfc: error function and ..... erf(3M)
error messages perror, errno, sys_errlist, sys_nerr: system ..... perror(3C)
error function and complementary error function ..... erf(3M)
error function erf, erfc: error function and complementary ..... erf(3M)
t_rcvuderr: receive a unit data error indication ..... t_rcvuderr(3N)
system diagnostic messages to form error log ..... dmesg(1M)
errorlog: internal error log file ..... errorlog(4)

strclean: STREAMS error logger cleanup program ..... strclean(1M)
strerr: STREAMS error logger daemon ..... strerr(1M)
log: interface to STREAMS error logging and event tracing ..... log(7)
t_error: produce error message ..... t_error(3N)
errormsg: error message text file ..... errormsg(4)
errno, sys_errlist, sys_nerr: system error messages ..... perror(3C)
introduction to system calls and error numbers ..... intro(2)
matherr: error-handling function ..... matherr(3M)
errorlog: internal error log file ..... errorlog(4)
errormsg: error message text file ..... errormsg(4)
spellin, hashcheck: find spelling errors ..... spell(1)
/etc/shadow with information from / etc/passwd ..... pwconv(1M)
pwconv: installs and updates / etc/shadow with information from/ ..... pwconv(1M)
end, etext, edata: last locations in program ..... end(3C)
SCED: SCSI/ Ethernet/Diagnostics board ..... sced(7)
hypot: Euclidean distance function ..... hypot(3M)
expr: evaluate arguments as an expression ..... expr(1)
test: condition evaluation command ..... test(1)
t_look: look at the current event on a transport endpoint ..... t_look(3N)
to STREAMS error logging and event tracing ..... log(7)
sulog: log of substitute user events ..... sulog(4)
(visual) display editor based on ex ..... vi(1)
edit: text editor (variant of ex for casual users) ..... edit(1)
ex: text editor ..... ex(1)
execlp, execvp: execute a file exec: exec: execl, execv, execle, execve, ... exec(2)
execlp, execvp: execute a/ exec: execl, execv, execle, execve, ... exec(2)
exec: execl, execv, execle, execve, execle, execve, execlp, execvp: execute a file ..... exec(2)
exec: execl, execv, execle, execve, execlp, execvp: execute a file ..... exec(2)
run: run an executable ..... run(1F)
execle, execve, execlp, execvp: execute a file ..... exec(2)
m_fork: execute a subprogram in parallel ..... m_fork(3PPS)
construct argument list(s) and execute command ..... xargs(1)
at, batch: execute commands at a later time ..... at(1)
regcmp, regex: compile and execute regular expression ..... regcmp(3X)
uuxqt: execute remote command requests ..... uuxqt(1M)
env: set environment for command execution ..... env(1)
suspend and resume child process execution ..... m_park_procs(3PPS)
/system command execution ..... uux(1C)
sleep: suspend execution for an interval ..... sleep(1)
sleep: suspend execution for interval ..... sleep(3C)
monitor: prepare execution profile ..... monitor(3C)
profil: execution time profile ..... profil(2)
execvp: execute a file exec: execl, execv, execle, execve, execlp, ... exec(2)
exec: execl, execv, execle, execve, execlp, execvp: execute a file ..... exec(2)
execv, execle, execve, execlp, execvp: execute a file ..... exec(2)
exit, _exit: terminate process ..... exit(2)
exit, _exit: terminate process ..... exit(2)
exponential, logarithm, power/ exp, log, log10, pow, sqrt: ..... exp(3M)
uncompress, zcat: compress and expand data ..... compress(1)
pack, pcat, unpack: compress and expand files ..... pack(1)
exp, log, log10, pow, sqrt: exponential, logarithm, power, square/ ..... exp(3M)
expression expr: evaluate arguments as an ..... expr(1)

expr: evaluate arguments as an expression ..... expr(1)
regex: compile and execute regular expression ..... regcmp(3X)
regcmp: regular expression compile ..... regcmp(1)
regexp: regular expression compile and match routines ..... regexp(5)
for a pattern using full regular expressions ..... egrep(1)
efl: Extended Fortran Language ..... efl(1)
edc: run an externally described command ..... edc(1M)
fsplit: split f77, ratfor, or efl files ..... fsplit(1)
floor, ceil, fmod, fabs: floor, ceiling, remainder, absolute value/ ..... floor(3M)
pathalias: alias file for FACE ..... alias(4)
.environ, .pref, .variables: FACE files ..... env(4)
report inter-process communication facilities status ..... ipcs(1)
a number factor: obtain the prime factors of ..... factor(1)
factor: obtain the prime factors of a number ..... factor(1)
true, false: provide truth values ..... true(1)
abort: generate an IOT fault ..... abort(3C)
stream fclose, fflush: close or flush a ..... fclose(3S)
fcntl: file control ..... fcntl(2)
fcntl: file control options ..... fcntl(5)
number to string ecvt, fcvt, gcvt: convert floating-point ..... ecvt(3C)
fopen, freopen, fdopen: open a stream ..... fopen(3S)
ferror, feof, clearerr, fileno: stream status inquiries ..... ferror(3S)
stream status inquiries ferror, feof, clearerr, fileno: ..... ferror(3S)
for a filesystem ff: list filenames and statistics ..... ff(1M)
fclose, fflush: close or flush a stream ..... fclose(3S)
from a stream getc, getchar, fgetc, getw: get character or word ..... getc(3S)
/getgrnam, setgrent, endgrent, fgetgrent: get group file entry ..... getgrent(3C)
/getpwnam, setpwent, endpwent, fgetpwent: get password file entry ..... getpwent(3C)
gets, fgets: get a string from a stream ..... gets(3S)
/getspnam, setspent, endspent, fgetspent, lckpwdf, ulckpwdf: get/ ..... getspent(3X)
string fgrep: search a file for a character ..... fgrep(1)
cut: cut out selected fields of each line of a file ..... cut(1)
mkfifo: make a FIFO special file ..... mkfifo(3C)
access: determine accessibility of a file ..... access(2)
chmod: change mode of file ..... chmod(2)
chown: change owner and group of a file ..... chown(2)
core: format of core image file ..... core(4)
cronlog: cron history file ..... cronlog(4)
crontab: user crontab file ..... crontab(1)
crontabs: clock daemon file ..... crontabs(4)
ctags: create a tags file ..... ctags(1)
selected fields of each line of a file ..... cut(1)
dd: convert and copy a file ..... dd(1M)
make a delta (change) to an SCCS file ..... delta(1)
errorlog: internal error log file ..... errorlog(4)
errormsg: error message text file ..... errormsg(4)
execve, execlp, execvp: execute a file ..... exec(2)
get: get a version of an SCCS file ..... get(1)
group: group file ..... group(4)
issue: issue identification file ..... issue(4)
header of a member of an archive file ..... ldahread(3X)
ldaclose: close a common object file ..... ldclose(3X)

the file header of a common object file.....ldfhread(3X)
of a section of a common object file.....ldlseek(3X)
file header of a common object file.....ldohseek(3X)
of a section of a common object file.....ldrseek(3X)
section header of a common object file.....ldshread(3X)
section of a common object file.....ldsseek(3X)
table entry of a common object file.....ldtbindex(3X)
table entry of a common object file.....ldtbread(3X)
the symbol table of a common object file.....ldtbseek(3X)
number entries in a common object file.....linenum(4)
link: link to a file.....link(2)
mkfifo: make a FIFO special file.....mkfifo(3C)
mknod: build special file.....mknod(1M)
directory, or a special or ordinary file.....mknod(2)
format of a module configuration file.....mod_config(4)
motd: message-of-the-day file.....motd(4)
newform: change the format of a text file.....newform(1)
null: the null file.....null(7)
passwd: password file.....passwd(4)
files or subsequent lines of one file.....paste(1)
product: installed products file.....product(4)
prs: print an SCCS file.....prs(1)
rcsfile: format of RCS file.....rcsfile(4)
read: read from file.....read(2)
readv: read vector from file.....readv(2SEQ)
reinit: run an initialization file.....reinit(1F)
information for a common object file.....reloc(4)
rename: change the name of a file.....rename(2)
rmdel: remove a delta from an SCCS file.....rmdel(1)
compare two versions of an SCCS file.....sccsdiff(1)
sccsfile: format of SCCS file.....sccsfile(4)
sccstorcs: build RCS file from SCCS file.....sccstorcs(1)
section header for a common object file.....scnhdr(4)
format of curses screen image file.....scr_dump(4)
shadow: restricted password file.....shadow(4)
system shutdown information file.....shutdown(4)
spellhist: unmatched spelling words file.....spellhist(4)
print checksum and block count of a file.....sum(1)
symlink: make symbolic link to a file.....symlink(2SEQ)
format of a system configuration file.....sys_config(4)
tail: deliver the last part of a file.....tail(1)
term: format of compiled term file.....term(4)
tmpfile: create a temporary file.....tmpfile(3S)
create a name for a temporary file.....tmpnam(3S)
access and modification times of a file.....touch(1)
undo a previous get of an SCCS file.....unget(1)
uniq: report repeated lines in a file.....uniq(1)
add/delete users in utmp file.....utadduser(3X)
the uucp directories and permissions file.....uucheck(1M)
val: validate SCCS file.....val(1)
write: write on a file.....write(2)
writev: write vector on a file.....writev(2SEQ)

utime: set file access and modification times.....utime(2)
ldfcn: common object file access routines.....ldfcn(4)
readfile, longline: reads file and gets longest line.....readfile(1F)
cpset: copy file and set mode, user, and group IDs.....cpset(1)
Hewlett-Packard 2645A terminal tape file archiver.....hpio(1)
tar: tape file archiver.....tar(1)
cpio: copy file archives in and out.....cpio(1)
rcs: change RCS file attributes.....rcs(1)
pwck, grpck: password/group file checkers.....pwck(1M)
mcs: manipulate the object file comment section.....mcs(1)
diff: differential file comparison.....diff(1)
diff3: 3-way differential file comparison.....diff3(1)
fcntl: file control.....fcntl(2)
fcntl: file control options.....fcntl(5)
UNIX-to-UNIX system file copy.....uuto(1C)
umask: set and get file creation mask.....umask(2)
close: close a file descriptor.....close(2)
dup: duplicate an open file descriptor.....dup(2)
dup2: duplicate an open file descriptor.....dup2(2)
file: determine file type.....file(1)
sact: print current SCCS file editing activity.....sact(1)
crypt: password and file encryption functions.....crypt(3X)
endgrent, fgetgrent: get group file entry.....getgrent(3C)
get file system descriptor file entry.....getmntent(3SEQ)
endpwent, fgetpwent: get password file entry.....getpwent(3C)
ulckpwdf: get shadow password file entry.....getspent(3X)
utmpname, getutpid: access utmp file entry.....getut(3C)
putpwent: write password file entry.....putpwent(3C)
putspent: write shadow password file entry.....putspent(3X)
fgrep: search a file for a character string.....fgrep(1)
grep: search a file for a pattern.....grep(1)
egrep: search a file for a pattern using full regular expressions.....egrep(1)
proto: prototype job file for at.....proto(4)
pathalias: alias file for FACE.....alias(4)
ldaopen: open a common object file for reading.....ldopen(3X)
acct: per-process accounting file format.....acct(4)
ar: common archive file format.....see *C and Lang Tools Binder*
manager, updates text files in ISIS file format.....isismgr(1M)
intro: introduction to file formats.....intro(4)
sccstorcs: build RCS file from SCCS file.....sccstorcs(1)
number entries of a common object file function.....ldlread(3X)
filehdr: file header for common object files.....filehdr(4)
limits: file header for implementation-specific/.....limits(4)
unistd: file header for symbolic constants.....unistd(4)
ldfhread: read the file header of a common object file.....ldfhread(3X)
ldohseek: seek to the optional file header of a common object file.....ldohseek(3X)
split: split a file into pieces.....split(1)
mmap: map an open file into the processs address space.....mmap(2SEQ)
ISIS file format isismgr: ISIS file manager, updates text files in.....isismgr(1M)
merge: three-way file merge.....merge(1)
ttyslot: find the slot in the utmp file of the current user.....ttyslot(3C)
more: view a file one full screen at a time.....more(1)

fuser: identify processes using a file or file structure.....fuser(1M)
creat: create a new file or rewrite an existing one.....creat(2)
pg: file perusal filter for CRTs.....pg(1)
lseek: move read/write file pointer.....lseek(2)
fseek, rewind, ftell: reposition a file pointer in a stream.....fseek(3S)
bfs: big file scanner.....bfs(1)
from a two to a one password file scheme.....pwunconv(1M)
stat, lstat, fstat: get file status.....stat(2)
identify processes using a file or file structure.....fuser(1M)
symbol name for common object file symbol table entry.....ldgetname(3X)
syms: common object file symbol table format.....syms(4)
newfs: construct a new file system.....newfs(1M)
/read directory entries and put in a file system independent format.....getdents(2)
/getmntent, hasmntopt, setmntent: get file system descriptor file entry.....getmntent(3SEQ)
dirent: file system independent directory entry.....dirent(4)
fs, inode: format of file system volume.....fs(4)
truncate, ftruncate: truncate a file to a specified length.....truncate(2SEQ)
uusched: the scheduler for the uucp file transport program.....uusched(1M)
uucico: file transport program for the uucp system.....uucico(1M)
ftw: walk a file tree.....ftw(3C)
file: determine file type.....file(1)
umask: set file-creation mode mask.....umask(1)
object files filehdr: file header for common.....filehdr(4)
mktemp: make a unique filename.....mktemp(3C)
search Interpreter criteria for filename.....pathconv(1F)
ctermid: generate filename for terminal.....ctermid(3S)
ff: list filenames and statistics for a filesystem.....ff(1M)
ferror, feof, clearerr, fileno: stream status inquiries.....ferror(3S)
search and print process accounting files.....acctcom(1)
merge or add total accounting files.....acctmerg(1M)
admin: create and administer SCCS files.....admin(1)
cat: concatenate and print files.....cat(1)
cmp: compare two files.....cmp(1)
or reject lines common to two sorted files.....comm(1)
config: build system configuration files.....config(1M)
cp, ln, mv: copy, link or move files.....cp(1)
.environ, .pref, .variables: FACE files.....env(4)
file header for common object files.....filehdr(4)
find: find files.....find(1)
fspec: format specification in text files.....fspec(4)
fsplit: split f77, ratfor, or efl files.....fsplit(1)
string, format of graphical files.....gps(4)
ident: identify files.....ident(1)
intro: introduction to special files.....intro(7)
lockf: record locking on files.....lockf(3C)
makedev: make system special files.....makedev(1M)
pcat, unpack: compress and expand files.....pack(1)
pr: print files.....pr(1)
and other information about RCS files.....rlog(1)
environment variables in core or in files.....set(1F)
sort: sort and/or merge files.....sort(1)
uucp: networking support files.....uucp(4)

what: identify SCCS files ..... what(1)
link, unlink: link and unlink files and directories ..... link(1M)
init.d: system level transition files directory ..... init.d(4)
ISIS file manager, updates text files in ISIS file format ..... isismgr(1M)
passmgmt: password files management ..... passmgmt(1M)
rm, rmdir: remove files or directories ..... rm(1)
paste: merge same lines of several files or subsequent lines of one/ ..... paste(1)
ott: files that hold object architecture information ..... ott(4)
checkfsys: check a filesystem ..... checkfsys(1M)
list filenames and statistics for a filesystem ..... ff(1M)
mkfs: construct a filesystem ..... mkfs(1M)
mount: mount a filesystem ..... mount(2)
repquota: summarize quotas for a filesystem ..... repquota(1M)
tunefs: tune up an existing filesystem ..... tunefs(1M)
umount: unmount a filesystem ..... umount(2)
volcopy: make literal copy of filesystem ..... volcopy(1M)
ckbupscd: check filesystem backup schedule ..... ckbupscd(1M)
interactive repair fsck: filesystem consistency check and ..... fsck(1M)
dcheck: filesystem directory consistency check ..... dcheck(1M)
dump: incremental filesystem dump ..... dump(1M)
fstyp: determine filesystem identifier ..... fstyp(1M)
dumpfs: dump filesystem information ..... dumpfs(1M)
statfs, fstatfs: get filesystem information ..... statfs(2)
quotacheck: filesystem quota consistency checker ..... quotacheck(1M)
quotaon, quotaoff: turn filesystem quotas on and off ..... quotaon(1M)
restore: incremental filesystem restore ..... restore(1M)
ustat: get filesystem statistics ..... ustat(2)
fsstat: report filesystem status ..... fsstat(1M)
icheck: filesystem storage consistency check ..... icheck(1M)
mnttab: mounted filesystem table ..... mnttab(4)
labelit: provide labels for filesystems ..... labelit(1M)
mount, umount: mount and unmount filesystems ..... mount(1M)
umountall: mount, unmount multiple filesystems ..... mountall(1M)
checklist: list of filesystems processed by fsck and ncheck ..... checklist(4)
fstab: file-system-table ..... fstab(4)
greek: select terminal filter ..... greek(1)
nl: line numbering filter ..... nl(1)
pg: file perusal filter for CRTs ..... pg(1)
col: filter reverse line-feeds ..... col(1)
lpfilter: administer filters used with the LP print service ..... lpfilter(1M)
find: find files ..... find(1)
for an object library lorder: find ordering relation ..... see *C and Lang Tools Binder*
display/change system boot flags ..... bootflags(1M)
/fpgetsticky, fpsetsticky: IEEE floating-point environment control ..... fpgetround(3C)
isnan: isnand, isnanf: test for floating-point NaN (Not-A-Number) ..... isnan(3C)
ecvt, fcvt, gcvt: convert floating-point number to string ..... ecvt(3C)
ldexp, modf: manipulate parts of floating-point numbers ..... frexp(3C)
ceiling, remainder, absolute value/ floor, ceil, fmod, fabs: floor, ... floor(3M)
floor, ceiling, remainder, absolute value/ ..... floor(3M)
fclose, fflush: close or flush a stream ..... fclose(3S)
a co-process with the controlling FMLI object ..... vsig(1F)
floor, ceil, fmod, fabs: floor, ceiling, remainder, absolute/ ..... floor(3M)

stream fopen, freopen, fdopen: open a ..... fopen(3S)
tcgetpgrp: get foreground process group ID..... tcgetpgrp(2)
tcsetpgrp: set foreground process group ID..... tcsetpgrp(2)
fork: create a new process..... fork(2)
system diagnostic messages to form error log..... dmesg(1M)
reset: reset the current form field to its default values ..... reset(1F)
acct: per-process accounting file format ..... acct(4)
dump, dumpdates: incremental dump format..... dump(4)
put in a file system independent format..... getdents(2)
updates text files in ISIS file format..... isismgr(1M)
common object file symbol table format..... syms(4)
management functions format: format disks or perform other defect..... format(1M)
other defect management functions format: format disks or perform ..... format(1M)
mod_config: format of a module configuration file ..... mod_config(4)
sys_config: format of a system configuration file..... sys_config(4)
newform: change the format of a text file..... newform(1)
term: format of compiled term file..... term(4)
core: format of core image file ..... core(4)
cpio: format of cpio archive..... cpio(4)
scr_dump: format of curses screen image file..... scr_dump(4)
dir: format of directories..... dir(4)
fs, inode: format of file system volume ..... fs(4)
gps: graphical primitive string, format of graphical files..... gps(4)
rcsfile: format of RCS file..... rcsfile(4)
sccsfile: format of SCCS file ..... sccsfile(4)
fspec: format specification in text files ..... fspec(4)
strftime: format time information ..... strftime(3C)
intro: introduction to file formats ..... intro(4)
utmp, wtmp: utmp and wtmp entry formats..... utmp(4)
scanf, fscanf, sscanf: convert formatted input ..... scanf(3S)
printf, fprintf, sprintf: print formatted output ..... printf(3S)
vprintf, vfprintf, vsprintf: print formatted output of a varargs/..... vprintf(3S)
pwfix: remove incorrectly formatted password entries..... pwfix(1M)
standalone embedded SCSI disk formatter..... ccsformat(8)
zdformat: standalone disk formatter for dual-channel disk/ ..... zdformat(8)
ASSIST menus or command forms..... astgen(1)
ec: forms compiler..... ec(1M)
lpforms: administer forms used with the LP print service..... lpforms(1M)
addformopt: add a form/SELECT window option to a menu ..... addformopt(1M)
ratfor: rational Fortran dialect..... ratfor(1)
efl: Extended Fortran Language ..... efl(1)
variables pathconf, fpathconf: get configurable pathname..... pathconf(2)
fpgetround, fpsetround, fpgetmask, fpgetsticky,/..... fpgetround(3C)
fpsetmask, fpgetsticky,/ fpgetround, fpsetround, fpgetmask, ... fpgetround(3C)
/fpsetround, fpgetmask, fpsetmask, fpgetsticky, fpsetsticky: IEEE/..... fpgetround(3C)
printf, fprintf, sprintf: print formatted output..... printf(3S)
fpgetround, fpsetround, fpgetmask, fpsetmask, fpgetsticky, fpsetsticky:/ ..... fpgetround(3C)
fpgetsticky,/ fpgetround, fpsetround, fpgetmask, fpsetmask, ... fpgetround(3C)
/fpgetmask, fpsetmask, fpgetsticky, fpsetsticky: IEEE floating-point/ ..... fpgetround(3C)
putc, putchar, fputc, putw: put character or word on a stream ..... putc(3S)
puts, fputs: put a string on a stream..... puts(3S)
getfrm: returns the current frame number..... getfrm(1F)

fread, fwrite: binary input/output ..... fread(3S)
t_free: free a library structure ..... t_free(3N)
df: report number of free disk blocks and inodes ..... df(1M)
malloc, free, realloc, calloc: main memory allocator ..... malloc(3C)
mallinfo: fast main memory/ malloc, free, realloc, calloc, mallopt, ... malloc(3X)
fopen, freopen, fdopen: open a stream ..... fopen(3S)
of floating-point numbers frexp, ldexp, modf: manipulate parts ..... frexp(3C)
gencc: create a front-end to the cc command ..... gencc(1M)
volume fs, inode: format of file system ..... fs(4)
scanf, fscanf, sscanf: convert formatted input ..... scanf(3S)
list of filesystems processed by fsck and ncheck ..... checklist(4)
and interactive repair fsck: filesystem consistency check ..... fsck(1M)
file pointer in a stream fseek, rewind, ftell: reposition a ..... fseek(3S)
generation numbers fsirand: install random inode ..... fsirand(1M)
files fspec: format specification in text ..... fspec(4)
files fsplit: split f77, ratfor, or efl ..... fsplit(1)
fsstat: report filesystem status ..... fsstat(1M)
fstab: file-system-table ..... fstab(4)
stat, lstat, fstat: get file status ..... stat(2)
statfs, fstatfs: get filesystem information ..... statfs(2)
identifier fstyp: determine filesystem ..... fstyp(1M)
fseek, rewind, ftell: reposition a file pointer in a stream ..... fseek(3S)
communication package stdipc: ftok: standard interprocess ..... stdipc(3C)
specified length truncate, ftruncate: truncate a file to a ..... truncate(2SEQ)
ftw: walk a file tree ..... ftw(3C)
function and complementary error function ..... erf(3M)
gamma: log gamma function ..... gamma(3M)
hypot: Euclidean distance function ..... hypot(3M)
entries of a common object file function ..... ldlread(3X)
matherr: error-handling function ..... matherr(3M)
prof: profile within a function ..... prof(5)
erf, erfc: error function and complementary error function ..... erf(3M)
j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
crypt: password and file encryption functions ..... crypt(3X)
logarithm, power, square root functions ..... exp(3M)
ceiling, remainder, absolute value functions ..... floor(3M)
or perform other defect management functions ..... format(1M)
sinh, cosh, tanh: hyperbolic functions ..... sinh(3M)
sysi86: machine specific functions ..... sysi86(2)
keys to internal menu system functions ..... syskeymap(4)
acos, atan, atan2: trigonometric functions ..... trig(3M)
math: math functions and constants ..... math(5)
intro: introduction to functions and libraries ..... intro(3)
300, 300s: handle special functions of DASI 300 and 300s terminals ..... 300(1)
hp: handle special functions of Hewlett-Packard terminals ..... hp(1)
450: handle special functions of the DASI 450 terminal ..... 450(1)
file or file structure fuser: identify processes using a ..... fuser(1M)
specified controlling terminal fvhangup: virtually hangup the ..... fvhangup(2SEQ)
fread, fwrite: binary input/output ..... fread(3S)
accounting records fwtmp, wtmpfix: manipulate connect ..... fwtmp(1M)
gamma: log gamma function ..... gamma(3M)
gamma: log gamma function ..... gamma(3M)

ecvt, fcvt, gcvt: convert floating-point number to string .....ecvt(3C)
command gencc: create a front-end to the cc.....gencc(1M)
command forms astgen: program for generating/modifying ASSIST menus or.....astgen(1)
fsirand: install random inode generation numbers.....fsirand(1M)
rand, srand: simple random-number generator .....rand(3C)
streamtty: Generic STREAMS tty driver interface.....streamtty(7)
tty: controlling terminal (gentty) interface .....tty(7)
getgeombyname: get disk geometry by name .....getgeombyname(3SEQ)
character or word from a stream getc, getchar, fgetc, getw: get.....getc(3S)
or word from a stream getc, getchar, fgetc, getw: get character .....getc(3S)
working directory getcwd: get pathname of current.....getcwd(3C)
put in a file system independent/ getdents: read directory entries and.....getdents(2)
descriptor table index getdtablemax: get max in-use.....getdtablemax(2SEQ)
size getdtablesize: get descriptor table.....getdtablesize(2SEQ)
getuid, geteuid, getgid, getegid: get real user, effective user, real/.....getuid(2)
name getenv: return value for environment.....getenv(3C)
user, effective user, real/ getuid, geteuid, getgid, getegid: get real.....getuid(2)
number getfrm: returns the current frame .....getfrm(1F)
geometry by name getgeombyname: get disk/.....getgeombyname(3SEQ)
getuid, geteuid, getgid, getegid: get real user, effective user,/.....getuid(2)
setgrent, endgrent, fgetgrent: get/ getgrent, getgrgid, getgrnam, ...getgrent(3C)
endgrent, fgetgrent: get/ getgrent, getgrgid, getgrnam, setgrent,.....getgrent(3C)
fgetgrent: get/ getgrent, getgrgid, getgrnam, setgrent, endgrent, ...getgrent(3C)
IDs getgroups: get supplementary group.....getgroups(2)
currently marked menu items getitems: returns a list of the.....getitems(1F)
getlogin: get login name .....getlogin(3C)
file system/ addmntent, endmntent, getmntent, hasmntopt, setmntent: get....getmntent(3SEQ)
stream getmsg: get next message off a .....getmsg(2)
argument vector getopt: get option letter from.....getopt(3C)
getopt: parse command options.....getopt(1)
getopts, getoptcvt: parse command options.....getopts(1)
options getopts, getoptcvt: parse command.....getopts(1)
getpagesize: get system page size.....getpagesize(2SEQ)
getpass: read a password.....getpass(3C)
process group, and parent/ getpid, getpgrp, getppid: get process, ...getpid(2)
process, process group, and parent/ getpid, getpgrp, getppid: get.....getpid(2)
and parent process/ getpid, getpgrp, getppid: get process, process group,.....getpid(2)
utilization information get_process_stats: get resource....get_process_stats(2SEQ)
getpseudotty: get pseudo terminals.....getpseudotty(3SEQ)
getpw: get name from IUD .....getpw(3C)
setpwent, endpwent, fgetpwent: get/ getpwent, getpwuid, getpwnam,.....getpwent(3C)
fgetpwent: get/ getpwent, getpwuid, getpwnam, setpwent, endpwent,.....getpwent(3C)
endpwent, fgetpwent: get/ getpwent, getpwuid, getpwnam, setpwent,.....getpwent(3C)
stream gets, fgets: get a string from a.....gets(3S)
readfile, longline: reads file and gets longest line.....readfile(1F)
scsiinfo structure getscsiinfo, getscsimatch: get .....getscsiinfo(3SEQ)
getscsiinfo, getscsimatch: get scsiinfo structure.....getscsiinfo(3SEQ)
endspent, fgetspent, lckpwdf,/ getspent, getspnam, setspent,.....getspent(3X)
fgetspent, lckpwdf,/ getspent, getspnam, setspent, endspent,.....getspent(3X)
speed and terminal settings used by getty.....gettydefs(4)
speed, and line discipline getty: set terminal type, modes,.....getty(1M)
ct: spawn getty to a remote terminal.....ct(1C)

settings used by getty gettydefs: speed and terminal .....gettydefs(4)
get real user, effective user, real/ getuid, geteuid, getgid, getegid:.....getuid(2)
microsecond clock getusclk, usclk_init: get/init.....getusclk(3SEQ)
pututline, setutent, endutent,/ getut: getutent, getutid, getutline,.....getut(3C)
pututline, setutent,/ getut: getutent, getutid, getutline,.....getut(3C)
getut: getutent, getutid, getutline, pututline, setutent,/.....getut(3C)
getut: getutent, getutid, getutline, pututline, setutent, endutent,/.....getut(3C)
/setutent, endutent, utmpname, getutpid: access utmp file entry.....getut(3C)
getc, getchar, fgetc, getw: get character or word from a stream .....getc(3S)
zdinfo structure getzdinfo, getzdinfobyname: get.....getzdinfo(3SEQ)
getzdinfo, getzdinfobyname: get zdinfo structure .....getzdinfo(3SEQ)
m_next: increment global counter .....m_next(3PPS)
task: global security table .....task(4)
operating system terms and symbols glossary: definitions of common.....glossary(1)
mktime, tzset:/ ctime, localtime, gmtime, asctime, cftime, ascftime,.....ctime(3C)
setjmp, longjmp: non-local goto.....setjmp(3C)
format of graphical files gps: graphical primitive string,.....gps(4)
primitive string, format of graphical files.....gps(4)
of graphical files gps: graphical primitive string, format.....gps(4)
plot: graphics interface .....plot(4)
plot: graphics interface subroutines.....plot(3X)
greek: select terminal filter .....greek(1)
grep: search a file for a pattern .....grep(1)
chown, chgrp: change owner or group.....chown(1)
newgrp: log in to a new group.....newgrp(1M)
run command in its own process group.....setpgrp(1)
initgroups: initialize group access list .....initgroups(3SEQ)
setgroups: set group access list .....setgroups(2SEQ)
/get real user, effective user, real group, and effective group IDs.....getuid(2)
/getppid: get process, process group, and parent process IDs .....getpid(2)
group: group file.....group(4)
pwck, grpck: password/ group file checkers.....pwck(1M)
setgrent, endgrent, fgetgrent: get group file entry .....getgrent(3C)
group: group file.....group(4)
setpgrp: set process group ID .....setpgrp(2)
create session and set process group ID .....setsid(2)
tcgetpgrp: get foreground process group ID.....tcgetpgrp(2)
tcsetpgrp: set foreground process group ID .....tcsetpgrp(2)
copy file and set mode, user, and group IDs.....cpset(1)
getgroups: get supplementary group IDs.....getgroups(2)
user, real group, and effective group IDs.....getuid(2)
setuid, setgid: set user and group IDs.....setuid(2)
id: print user and group IDs and names.....id(1M)
groups: show group memberships .....groups(1)
chown: change owner and group of a file.....chown(2)
send a signal to a process or a group of processes.....kill(2)
maintain, update, and regenerate groups of programs .....make(1)
groups: show group memberships.....groups(1)
pwck, grpck: password/group file checkers.....pwck(1M)
ssignal, gsignal: software signals.....ssignal(3C)
stty, gtty: set and get terminal state (defunct).....stty(2)
terminal fvhangup: virtually hangup the specified controlling.....fvhangup(2SEQ)

nohup: run a command immune to ... hangups and quits ... nohup(1)
hsearch, hcreate, hdestroy: manage ... hash search tables ... hsearch(3C)
spell, hashmake, spellin, ... hashcheck: find spelling errors ... spell(1)
crypt, setkey, encrypt: generate ... hashing encryption ... crypt(3C)
spelling errors spell, ... hashmake, spellin, hashcheck: find ... spell(1)
addmntent, endmntent, getmntent, ... hasmntopt, setmntent: get file/ ... getmntent(3SEQ)
search tables hsearch, ... hcreate, hdestroy: manage hash ... hsearch(3C)
hsearch, hcreate, ... hdestroy: manage hash search tables ... hsearch(3C)
scnhdr: section ... header for a common object file ... scnhdr(4)
filehdr: file ... header for common object files ... filehdr(4)
limits: file ... header for implementation-specific constants ... limits(4)
unistd: file ... header for symbolic constants ... unistd(4)
ldfhread: read the file ... header of a common object file ... ldfhread(3X)
ldohseek: seek to the optional file ... header of a common object file ... ldohseek(3X)
/read an indexednamed section ... header of a common object file ... ldshread(3X)
ldahread: read the archive ... header of a member of an archive file ... ldahread(3X)
helplog: monitor ... help actions ... helplog(4)
help: operating system ... Help Facility ... help(1)
helpadm: change the ... Help Facility database ... helpadm(1M)
database ... help: operating system Help Facility ... help(1)
... helpadm: change the Help Facility ... helpadm(1M)
... helplog: monitor help actions ... helplog(4)
file archiver hpio: ... Hewlett-Packard 2645A terminal tape ... hpio(1)
hp: handle special functions of ... Hewlett-Packard terminals ... hp(1)
cronlog: cron ... history file ... cronlog(4)
ott: files that ... hold object architecture information ... ott(4)
uptime: show ... how long system has been up ... uptime(1)
Hewlett-Packard terminals ... hp: handle special functions of ... hp(1)
tape file archiver ... hpio: Hewlett-Packard 2645A terminal ... hpio(1)
hash search tables ... hsearch, hcreate, hdestroy: manage ... hsearch(3C)
sinh, cosh, tanh: ... hyperbolic functions ... sinh(3M)
... hypot: Euclidean distance function ... hypot(3M)
u3b5, vax: get processor/ machid: ... i386, ns32000, pdp11, u3b, u3b2, ... machid(1)
consistency check ... icheck: filesystem storage ... icheck(1M)
... ident: identify files ... ident(1)
m_get_myid: return process ... identification ... m_get_myid(3PPS)
issue: issue ... identification file ... issue(4)
fstyp: determine filesystem ... identifier ... fstyp(1M)
shmget: get shared memory segment ... identifier ... shmget(2)
file and set mode, user, and group ... IDs ... cpset(1)
getgroups: get supplementary group ... IDs ... getgroups(2)
process group, and parent process ... IDs ... getpid(2)
real group, and effective group ... IDs ... getuid(2)
setuid, setgid: set user and group ... IDs ... setuid(2)
id: print user and group ... IDs and names ... id(1M)
/fpsetmask, fpgetsticky, fpsetsticky: ... IEEE floating-point environment/ ... fpgetround(3C)
core: format of core ... image file ... core(4)
scr_dump: format of curses screen ... image file ... scr_dump(4)
crash: examine system ... images ... crash(1M)
limits: file header for ... implementation-specific constants ... limits(4)
pwfix: remove ... incorrectly formatted password entries ... pwfix(1M)
m_next: ... increment global counter ... m_next(3PPS)

dump, dumpdates: incremental dump format.....dump(4)
dump: incremental filesystem dump.....dump(1M)
restore: incremental filesystem restore.....restore(1M)
get maximum in-use descriptor table index.....getdtablemax(2SEQ)
common/ ldtbindex: compute the index of a symbol table entry of a.....ldtbindex(3X)
common object/ ldtbread: read an indexed symbol table entry of a.....ldtbread(3X)
common/ldshread, ldnshread: read an indexednamed section header of a.....ldshread(3X)
ldsseek, ldnsseek: seek to an indexednamed section of a common/.....ldsseek(3X)
receipt of an orderly release indication.....t_rcvrel(3N)
receive a unit data error indication.....t_rcvuderr(3N)
specific alarms and/or the working indicator.....indicator(1F)
specific alarms and/or the working/ indicator: display application.....indicator(1F)
calculate number of blocks and indirect blocks.....nblocks(3SEQ)
syscall: indirect system call.....syscall(2)
terminfo descriptions infocmp: compare or print out.....infocmp(1M)
getusclk, usclk_init: get/ init microsecond clock.....getusclk(3SEQ)
inittab: script for the init process.....inittab(4)
initialization init, telinit: process control.....init(1M)
files directory init.d: system level transition.....init.d(4)
list initgroups: initialize group access.....initgroups(3SEQ)
init, telinit: process control initialization.....init(1M)
reinit: run an initialization file.....reinit(1F)
bcheckrc: system initialization procedures.....bcheckrc(1M)
tput: initialize a terminal or query terminfo database.....tput(1)
s_init_barrier, s_wait_barrier: initialize barrier, wait at barrier.....s_wait_barrier(3PPS)
initgroups: initialize group access list.....initgroups(3SEQ)
/s_lock, s_clock, s_unlock: initialize, lock, unlock locks.....s_lock(3PPS)
setup: initialize system for first user.....setup(1)
inittab: script for the init process.....inittab(4)
clri: clear inode.....clri(1M)
fs, inode: format of file system volume.....fs(4)
fsirand: install random inode generation numbers.....fsirand(1M)
number of free disk blocks and inodes.....df(1M)
fscanf, sscanf: convert formatted input.....scanf(3S)
ungetc: push character back into input stream.....ungetc(3S)
fread, fwrite: binary input/output.....fread(3S)
poll: STREAMS input/output multiplexing.....poll(2)
stdio: standard buffered input/output package.....stdio(3S)
uustat: uucp status inquiry and job control.....uustat(1C)
install: install commands.....install(1M)
install: install commands.....install(1M)
fsirand: install random inode generation numbers.....fsirand(1M)
on the BOM bomverify: perform an installation consistency check based.....bomverify(1M)
on a disk installboot: installs a boot program.....installboot(1M)
product: installed products file.....product(4)
installboot: installs a boot program on a disk.....installboot(1M)
with information from/ pwconv: installs and updates /etc/shadow.....pwconv(1M)
dis386: disassemble Series 386 instructions.....dis386(3SEQ)
mailx: interactive message processing system.....mailx(1)
filesystem consistency check and interactive repair.....fsck(1M)
console: console interface.....console(7)
lp: Systech parallel line printer interface.....lp(7)

mtio: UNIX magnetic tape interface.....mtio(7)
plot: graphics interface.....plot(4)
Generic STREAMS tty driver interface.....streamtty(7)
swap: swap administrative interface.....swap(1M)
termio: general terminal interface.....termio(7)
tg: SCSI 1/2-inch magnetic tape interface.....tg(7)
tty: controlling terminal (gentty) interface.....tty(7)
wp: SSM parallel line printer interface.....wp(7)
Xylogics 472 MULITBUS magnetic tape interface.....xt(7)
timod: Transport Interface cooperating STREAMS module.....timod(7)
STREAMS module tirdwr: Transport Interface read/write.....tirdwr(7)
Transport Interface read/write interface STREAMS module.....tirdwr(7)
plot: graphics interface subroutines.....plot(3X)
sysadm: menu interface to do system administration.....sysadm(1)
and event tracing log: interface to STREAMS error logging.....log(7)
errorlog: internal error log file.....errorlog(4)
syskeymap: identify keys to internal menu system functions.....syskeymap(4)
asa: interpret ASA carriage control characters.....asa(1)
pathconv: search Interpreter criteria for filename.....pathconv(1F)
csh: a shell (command interpreter) with C-like syntax.....csh(1)
pipe: create an interprocess channel.....pipe(2)
ipcs: report inter-process communication facilities status.....ipcs(1)
stdipc: ftok: standard interprocess communication package.....stdipc(3C)
tfipc: interprocess communication reporter.....tfipc(1M)
ncheck: generate pathnames from i-numbers.....ncheck(1M)
streamio: STREAMS ioctl commands.....streamio(7)
ioctl: control device.....ioctl(2)
abort: generate an IOT fault.....abort(3C)
semaphore set, or shared memory ID ipcrm: remove a message queue,.....ipcrm(1)
communication facilities status ipcs: report inter-process.....ipcs(1)
/isxdigit, islower, isupper, isalpha, isalnum, isspace, iscntrl, ispunct,/.....ctype(3C)
isdigit, isxdigit, islower, isupper, isalpha, isalnum, isspace, iscntrl,/.....ctype(3C)
/iscntrl, ispunct, isprint, isgraph, isascii, tolower, toupper, toascci,/.....ctype(3C)
ttyname, isatty: find name of a terminal.....ttyname(3C)
/isupper, isalpha, isalnum, isspace, iscntrl, ispunct, isprint, isgraph,/.....ctype(3C)
isalpha, isalnum, isspace, iscntrl,/ isdigit, isxdigit, islower, isupper,.....ctype(3C)
/isspace, iscntrl, ispunct, isprint, isgraph, isascii, tolower, toupper,/.....ctype(3C)
file manager, updates text files in ISIS file format.....isismgr(1M)
files in ISIS file format isismgr: ISIS file manager, updates text.....isismgr(1M)
text files in ISIS file format isismgr: ISIS file manager, updates.....isismgr(1M)
isdigit, isxdigit, islower, isupper, isalpha, isalnum, isspace,/.....ctype(3C)
floating-point NaN (Not-A-Number) isnan: isnan, isnanf: test for.....isnan(3C)
isnan: isnand, isnanf: test for floating-point NaN/.....isnan(3C)
(Not-A-Number) isnan: isnand, isnanf: test for floating-point NaN.....isnan(3C)
/isalnum, isspace, iscntrl, ispunct, isprint, isgraph, isascii, tolower,/.....ctype(3C)
/isalpha, isalnum, isspace, iscntrl, ispunct, isprint, isgraph, isascii,/.....ctype(3C)
/islower, isupper, isalpha, isalnum, isspace, iscntrl, ispunct, isprint,/.....ctype(3C)
issue identification file.....issue(4)
isdigit, isxdigit, islower, isupper, isalpha, isalnum, isspace,/.....ctype(3C)
isalnum, isspace, iscntrl,/ isdigit, isxdigit, islower, isupper, isalpha,.....ctype(3C)
getpw: get name from IUD.....getpw(3C)
bessel: j0, j1, jn, y0, y1, yn: Bessel functions.....bessel(3M)

bessel: j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
bessel: j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
uustat: uucp status inquiry and job control ..... uustat(1C)
proto: prototype job file for at ..... proto(4)
join: relational database operator ..... join(1)
/erand48, lrand48, nrand48, mrand48, jrand48, srand48, seed48, lcong48:/ ..... drand48(3C)
sigsetjmp, siglongjmp: non-local jumps ..... sigsetjmp(3C)
dumpconf: dump kernel auto-configuration device table ..... dumpconf(1M)
makekey: generate encryption key ..... makekey(1)
syskeymap: identify keys to internal menu system functions ..... syskeymap(4)
apropos: locate commands by keyword lookup ..... apropos(1)
system command locate: use keywords to identify an operating ..... locate(1)
killall: kill all active processes ..... killall(1M)
m_kill_procs: kill child processes ..... m_kill_procs(3PPS)
a group of processes kill: send a signal to a process or ..... kill(2)
kill: terminate a process ..... kill(1)
killall: kill all active processes ..... killall(1M)
kMBmem, kMWmem: MULITBUS memory ..... kmmem(7)
mem, kmem: core memory ..... mem(7)
kMBmem, kMWmem: MULITBUS memory ..... kmmem(7)
ksh, rksh: KornShell, a standard/restricted command and /..... ksh(1)
standard/restricted command and/ ksh, rksh: KornShell, a ..... ksh(1)
integers and long integers l3tol, ltol3: convert between 3-byte ..... l3tol(3C)
and base-64 ASCII string a64l, l64a: convert between long integer ..... a64l(3C)
filesystems labelit: provide labels for ..... labelit(1M)
labelit: provide labels for filesystems ..... labelit(1M)
awk: pattern scanning and processing language ..... awk(1)
bc: arbitrary-precision arithmetic language ..... bc(1)
efl: Extended Fortran Language ..... efl(1)
command and programming language ..... ksh(1)
pattern scanning and processing language ..... nawk(1)
command programming language ..... sh(1)
cftime: language specific strings ..... cftime(4)
chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp,/ ..... acctsh(1M)
/setspent, endspent, fgetspent, lckpwdf, ulckpwdf: get shadow/ ..... getspent(3X)
/mrand48, jrand48, srand48, seed48, lcong48: generate uniformly/ ..... drand48(3C)
files ld: link editor for object ..... see *C and Lang Tools Binder*
ld0: STREAMS tty line discipline ..... ld0(7)
ldclose, ldaclose: close a common object file ..... ldclose(3X)
a member of an archive file ldahread: read the archive header of ..... ldahread(3X)
for reading ldopen, ldaopen: open a common object file ..... ldopen(3X)
object file ldclose, ldaclose: close a common ..... ldclose(3X)
floating-point numbers frexp, ldexp, modf: manipulate parts of ..... frexp(3C)
routines ldfcn: common object file access ..... ldfcn(4)
common object file ldfhread: read the file header of a ..... ldfhread(3X)
common object file symbol table/ ldgetname: retrieve symbol name for ..... ldgetname(3X)
number entries of a common/ldlread, ldlinit, ldlitem: manipulate line ..... ldlread(3X)
entries of a/ ldlread, ldlinit, ldlitem: manipulate line number ..... ldlread(3X)
manipulate line number entries of a/ ldlread, ldlinit, ldlitem: ..... ldlread(3X)
number entries of a section of a/ ldlseek, ldlseek: seek to line ..... ldlseek(3X)
entries of a section of a/ ldlseek, ldnlseek: seek to line number ..... ldlseek(3X)
of a section of a common/ ldrseek, ldnrseek: seek to relocation entries ..... ldrseek(3X)

section header of a/ ldshread, ldnshread: read an indexednamed.....ldshread(3X)
section of a common object/ ldsseek, ldnsseek: seek to an indexednamed.....ldsseek(3X)
header of a common object file ldohseek: seek to the optional file.....ldohseek(3X)
object file for reading ldopen, ldaopen: open a common.....ldopen(3X)
relocation entries of a section of/ ldrseek, ldnrseek: seek to.....ldrseek(3X)
indexednamed section header of a/ ldshread, ldnshread: read an.....ldshread(3X)
indexednamed section of a common/ ldsseek, ldnsseek: seek to an.....ldsseek(3X)
symbol table entry of a common/ ldtbindex: compute the index of a.....ldtbindex(3X)
table entry of a common object file ldtbread: read an indexed symbol.....ldtbread(3X)
of a common object file ldtbseek: seek to the symbol table.....ldtbseek(3X)
truncate a file to a specified length.....truncate(2SEQ)
init.d: system level transition files directory.....init.d(4)
lexical tasks lex: generate programs for simple.....lex(1)
lex: generate programs for simple lexical tasks.....lex(1)
lsearch, lfind: linear search and update.....lsearch(3C)
intro: introduction to functions and libraries.....intro(3)
find ordering relation for an object library.....see *C and Lang Tools Binder*
introduction to Parallel Programming Library.....intro(3PPS)
t_sync: synchronize transport library.....t_sync(3N)
t_alloc: allocate a library structure.....t_alloc(3N)
t_free: free a library structure.....t_free(3N)
shell (command interpreter) with C- like syntax.....csh(1)
quota: display disk usage and limits.....quota(1)
ulimit: get and set user limits.....ulimit(2)
implementation-specific constants limits: file header for.....limits(4)
returns the number of CPUs on- line.....cpus_online(3PPS)
line: read one line.....line(1)
puts its arguments on message line.....message(1F)
all but one active processor off line.....offline_all(3SEQ)
reads file and gets longest line.....readfile(1F)
establish an out-going terminal line connection.....dial(3C)
set terminal type, modes, speed, and line discipline.....getty(1M)
ld0: STREAMS tty line discipline.....ld0(7)
set terminal type, modes, speed, and line discipline.....uugetty(1M)
man: display on- line man pages.....man(1)
linenum: line number entries in a common object file.....linenum(4)
/ldlinit, ldlitem: manipulate line number entries of a common object/.....ldlread(3X)
a common/ ldlseek, ldnlseek: seek to line number entries of a section of.....ldlseek(3X)
nl: line numbering filter.....nl(1)
cut: cut out selected fields of each line of a file.....cut(1)
lp: Systech parallel line printer interface.....lp(7)
wp: SSM parallel line printer interface.....wp(7)
line: read one line.....line(1)
lsearch, lfind: linear search and update.....lsearch(3C)
col: filter reverse line-feeds.....col(1)
common object file linenum: line number entries in a.....linenum(4)
readlink: read value of a symbolic link.....readlink(2SEQ)
link, unlink: link and unlink files and directories.....link(1M)
ld: link editor for object files.....see *C and Lang Tools Binder*
a.out: common assembler and link editor output.....a.out(4)
link: link to a file.....link(2)
cp, ln, mv: copy, link or move files.....cp(1)

link: link to a file ..... link(2)
symlink: make symbolic link to a file ..... symlink(2SEQ)
and directories link, unlink: link and unlink files ..... link(1M)
lint: a C program checker ..... see *C and Lang Tools Binder*
initgroups: initialize group access list ..... initgroups(3SEQ)
nlist: get entries from name list ..... nlist(3C)
setgroups: set group access list ..... setgroups(2SEQ)
varargs: handle variable argument list ..... varargs(5)
output of a varargs argument list ..... vprintf(3S)
ls: list contents of directory ..... ls(1)
ff: list filenames and statistics for a filesystem ..... ff(1M)
fsck and ncheck checklist: list of filesystems processed by ..... checklist(4)
getitems: returns a list of the currently marked menu items ..... getitems(1F)
t_listen: listen for a connect request ..... t_listen(3N)
listing list: produce C source ..... see *C and Lang Tools Binder*
xargs: construct argument list(s) and execute command ..... xargs(1)
volcopy: make literal copy of filesystem ..... volcopy(1M)
cp, ln, mv: copy, link or move files ..... cp(1)
ascftime, mktime, tzset:/ ctime, localtime, gmtime, asctime, cftime, ... ctime(3C)
apropos: locate commands by keyword lookup ..... apropos(1)
operating system command locate: use keywords to identify an ..... locate(1)
setlocale: set programs location ..... setlocale(3C)
end, etext, edata: last locations in program ..... end(3C)
plock: lock process, text, or data in memory ..... plock(2)
m_lock, m_unlock: lock, unlock locks ..... m_lock(3PPS)
s_clock, s_unlock: initialize, lock, unlock locks ..... s_lock(3PPS)
lockf: record locking on files ..... lockf(3C)
lockf: record locking on files ..... lockf(3C)
m_lock, m_unlock: lock, unlock locks ..... m_lock(3PPS)
s_unlock: initialize, lock, unlock locks ..... s_lock(3PPS)
diagnostic messages to form error log ..... dmesg(1M)
errorlog: internal error log file ..... errorlog(4)
gamma: log gamma function ..... gamma(3M)
newgrp: log in to a new group ..... newgrp(1M)
logging and event tracing log: interface to STREAMS error ..... log(7)
logarithm, power, square root/ exp, log, log10, pow, sqrt: exponential, ... exp(3M)
about RCS files rlog: print log messages and other information ..... rlog(1)
/usr/adm/loginlog: log of failed login attempts ..... loginlog(4)
sulog: log of substitute user events ..... sulog(4)
logarithm, power, square/ exp, log, log10, pow, sqrt: exponential, ... exp(3M)
log, log10, pow, sqrt: exponential, logarithm, power, square root/ ..... exp(3M)
strclean: STREAMS error logger cleanup program ..... strclean(1M)
strerr: STREAMS error logger daemon ..... strerr(1M)
log: interface to STREAMS error logging and event tracing ..... log(7)
/usr/adm/loginlog: log of failed login attempts ..... loginlog(4)
getlogin: get login name ..... getlogin(3C)
logname: get login name ..... logname(1)
cuserid: get character login name of the user ..... cuserid(3S)
logname: return login name of user ..... logname(3X)
passwd: change login password and password attributes ..... passwd(1)
tfmenush: assign menu system as user login shell ..... tfmenush(1M)
login: sign on ..... login(1)

setting up an environment at login time.....profile(4)
/usr/adm/ loginlog: log of failed login attempts.....loginlog(4)
logname: get login name.....logname(1)
logname: return login name of user.....logname(3X)
setjmp, longjmp: non-local goto.....setjmp(3C)
readfile, longline: reads file and gets longest line.....readfile(1F)
apropos: locate commands by keyword lookup.....apropos(1)
make a database for fast password lookup.....mkpwdbm(1M)
relation for an object library lorder: find ordering.....see *C and Lang Tools Binder*
the LP print service lp, cancel: send/cancel requests to.....lp(1)
cancel: send/cancel requests to the LP print service.....lp(1)
lpadmin: configure the LP print service.....lpadmin(1M)
administer filters used with the LP print service.....lpfilter(1M)
administer forms used with the LP print service.....lpforms(1M)
information about the status of the LP print service.....lpstat(1)
/lpshut, lpmove: start/stop the LP print service and move requests.....lpsched(1M)
enable, disable: enable/disable LP printers.....enable(1)
accept, reject: allow or prevent LP requests.....accept(1M)
interface lp: Systech parallel line printer.....lp(7)
service lpadmin: configure the LP print.....lpadmin(1M)
with the LP print service lpfilter: administer filters used.....lpfilter(1M)
the LP print service lpforms: administer forms used with.....lpforms(1M)
processing module lpmod: parallel printer canonical.....lpmod(7)
service and move/ lpsched, lpshut, lpmove: start/stop the LP print.....lpsched(1M)
the LP print service and move/ lpsched, lpshut, lpmove: start/stop.....lpsched(1M)
print service and move/ lpsched, lpshut, lpmove: start/stop the LP.....lpsched(1M)
status of the LP print service lpstat: print information about the.....lpstat(1)
supported by the menu system lptypes: table of printer types.....lptypes(4)
priorities lpusers: set printing queue.....lpusers(1M)
srand48, seed48,/ drand48, erand48, lrand48, nrand48, mrand48, jrand48,.....drand48(3C)
ls: list contents of directory.....ls(1)
update lsearch, lfind: linear search and.....lsearch(3C)
lseek: move read/write file pointer.....lseek(2)
stat, lstat, fstat: get file status.....stat(2)
integers and long integers l3tol, ltol3: convert between 3-byte.....l3tol(3C)
m4: macro processor.....m4(1)
u3b2, u3b5, vax: get processor type/ machid: i386, ns32000, pdp11, u3b,.....machid(1)
show the configuration of the machine.....showcfg(1M)
sysi86: machine specific functions.....sysi86(2)
values: machine-dependent values.....values(5)
sgetl: access long integer data in a machine-independent fashion.....sputl(3X)
m4: macro processor.....m4(1)
mtio: UNIX magnetic tape interface.....mtio(7)
tg: SCSI 1/2-inch magnetic tape interface.....tg(7)
xt: Xylogics 472 MULITBUS magnetic tape interface.....xt(7)
mt: magnetic tape manipulating program.....mt(1)
rmail: send mail to users or read mail.....mail(1)
read mail mail, rmail: send mail to users or.....mail(1)
mail, rmail: send mail to users or read mail.....mail(1)
enhancement request mailbug: submit a bug report or.....mailbug(1)
processing system mailx: interactive message.....mailx(1)
groups of programs make: maintain, update, and regenerate.....make(1)

programs intro: introduction to ..... intro(1)
/ttyname_to_dev: convert between major/minor numbers/tty names ..... dev_to_ttyname(3C)
regenerate groups of programs make: maintain, update, and ..... make(1)
man page databases makealiases, makewhatis: generate ..... makealiases(1M)
makedev: make system special files ..... makedev(1M)
makekey: generate encryption key ..... makekey(1)
databases makealiases, makewhatis: generate man page ..... makealiases(1M)
/free, realloc, calloc, mallopt, mallinfo: fast main memory allocator ..... malloc(3X)
memory allocator malloc, free, realloc, calloc: main ..... malloc(3C)
mallopt, mallinfo: fast main memory/ malloc, free, realloc, calloc, ... malloc(3X)
malloc, free, realloc, calloc, mallopt, mallinfo: fast main memory/ ..... malloc(3X)
man: display on-line man pages ..... man(1)
makealiases, makewhatis: generate man page databases ..... makealiases(1M)
man: display on-line man pages ..... man(1)
tsearch, tfind, tdelete, twalk: manage binary search trees ..... tsearch(3C)
hsearch, hcreate, hdestroy: manage hash search tables ..... hsearch(3C)
endpoint t_optmgmt: manage options for a transport ..... t_optmgmt(3N)
passmgmt: password files management ..... passmgmt(1M)
sigignore, sigpause: signal management ..... sigset(2)
format disks or perform other defect management functions ..... format(1M)
file format isismgr: ISIS file manager, updates text files in ISIS ..... isismgr(1M)
mt: magnetic tape manipulating program ..... mt(1)
parameters of virtual-to-physical mapping devices ..... pmap(1M)
m_single, m_multi: mark single-process code section ..... m_single(3PPS)
umask: set file-creation mode mask ..... umask(1)
umask: set and get file creation mask ..... umask(2)
regex: match patterns against a string ..... regex(1F)
regular expression compile and match routines ..... regexp(5)
math: math functions and constants ..... math(5)
math: math functions and constants ..... math(5)
matherr: error-handling function ..... matherr(3M)
mbad: MULITBUS adapter ..... mbad(7)
comment section mcs: manipulate the object file ..... mcs(1)
mem, kmem: core memory ..... mem(7)
read the archive header of a member of an archive file ..... ldahread(3X)
groups: show group memberships ..... groups(1)
memset: memory operationsmemory: memccpy, memchr, memcmp, memcpy, ... memory(3C)
memory operationmemory: memccpy, memchr, memcmp, memcpy, memset: ..... memory(3C)
operationmemory: memccpy, memchr, memcmp, memcpy, memset: memory ..... memory(3C)
memory: memccpy, memchr, memcmp, memcpy, memset: memory operations ..... memory(3C)
kMBmem, kMWmem: MULITBUS memory ..... kmmem(7)
mem, kmem: core memory ..... mem(7)
lock process, text, or data in memory ..... plock(2)
create security table in shared memory ..... tasktbl(1M)
malloc, free, realloc, calloc: main memory allocator ..... malloc(3C)
calloc, mallopt, mallinfo: fast main memory allocator ..... malloc(3X)
shmalloc, shrealloc, shfree: shared memory allocator ..... shmalloc(3PPS)
valloc: aligned memory allocator ..... valloc(3SEQ)
shmctl: shared memory control operations ..... shmctl(2)
consmem, sm, ss: console memory drivers ..... consmem(7)
dump: standalone memory dump ..... dump(8)
spawn new process in a virtual memory efficient way ..... vfork(2)

queue, semaphore set, or shared memory ID ..... ipcrm(1)
memcpy, memset: memory operations memory: memccpy, memchr, memcmp, ... memory(3C)
memchr, memcmp, memcpy, memset: memory operations ..... memory(3C)
shmop, shmat, shmdt: shared memory operations ..... shmop(2)
vmtune: tune system-wide virtual- memory parameters ..... vmtune(1M)
shmget: get shared memory segment identifier ..... shmget(2)
vm_ctl: examine and change virtual memory tuning parameters ..... vm_ctl(2SEQ)
memccpy, memchr, memcmp, memcpy, memset: memory operations ..... memory(3C)
addcmdopt: add a command option to a menu ..... addcmdopt(1M)
add a form/SELECT window option to a menu ..... addformopt(1M)
addmenuopt: add a menu option to a menu ..... addmenuopt(1M)
sysadm: menu interface to do system administration ..... sysadm(1)
a list of the currently marked menu items ..... getitems(1F)
menu: menu processor ..... menu(1M)
menumsg: route user message to menu message queue ..... menumsg(1M)
rmvopt: remove a menu option ..... rmvopt(1M)
addmenuopt: add a menu option to a menu ..... addmenuopt(1M)
menu: menu processor ..... menu(1M)
of printer types supported by the menu system ..... lptypes(4)
tfmenush: assign menu system as user login shell ..... tfmenush(1M)
syskeymap: identify keys to internal menu system functions ..... syskeymap(4)
message queue menumsg: route user message to menu ..... menumsg(1M)
for generating/modifying ASSIST menus or command forms ..... astgen(1)
merge: three-way file merge ..... merge(1)
sort: sort and/or merge files ..... sort(1)
acctmerg: merge or add total accounting files ..... acctmerg(1M)
rcsmerge: merge RCS revisions ..... rcsmerge(1)
subsequent lines of one file paste: merge same lines of several files or ..... paste(1)
merge: three-way file merge ..... merge(1)
mesg: permit or deny messages ..... mesg(1)
t_error: produce error message ..... t_error(3N)
tfnotify: send completion message ..... tfnotify(1M)
msgctl: message control operations ..... msgctl(2)
message: puts its arguments on message line ..... message(1F)
getmsg: get next message off a stream ..... getmsg(2)
putmsg: send a message on a stream ..... putmsg(2)
msgop: message operations ..... msgop(2)
mailx: interactive message processing system ..... mailx(1)
message line message: puts its arguments on ..... message(1F)
menumsg: route user message to menu message queue ..... menumsg(1M)
msgget: get message queue ..... msgget(2)
shared memory ID ipcrm: remove a message queue, semaphore set, or ..... ipcrm(1)
errormsg: error message text file ..... errormsg(4)
menumsg: route user message to menu message queue ..... menumsg(1M)
motd: message-of-the-day file ..... motd(4)
mesg: permit or deny messages ..... mesg(1)
sys_errlist, sys_nerr: system error messages ..... perror(3C)
strace: print STREAMS trace messages ..... strace(1M)
rlog: print log messages and other information about RCS files ..... rlog(1)
dmesg: collect system diagnostic messages to form error log ..... dmesg(1M)
parallel m_fork: execute a subprogram in ..... m_fork(3PPS)
identification m_get_myid: return process ..... m_get_myid(3PPS)

number of child processes m_get_numprocs: return ..... m_get_numprocs(3PPS)
getusclk, usclk_init: get/init microsecond clock ..... getusclk(3SEQ)
usclk: 32-bit microsecond counter ..... usclk(7)
aging, PFF, swapping, or processor migration disabled ..... noage(1M)
clone: open any minor device on a STREAMS driver ..... clone(7)
/convert between major/ minor numbers and tty names ..... dev_to_ttyname(3C)
/acctwtmp: overview of accounting and miscellaneous accounting commands ..... acct(1M)
intro: introduction to miscellany ..... intro(5)
mkdir: make a directory ..... mkdir(2)
mkdir: make directories ..... mkdir(1)
mkfifo: make a FIFO special file ..... mkfifo(3C)
mkfs: construct a filesystem ..... mkfs(1M)
m_kill_procs: kill child processes ..... m_kill_procs(3PPS)
mknod: build special file ..... mknod(1M)
special or ordinary file mknod: make a directory, or a ..... mknod(2)
password lookup mkpwdbm: make a database for fast ..... mkpwdbm(1M)
mktemp: make a unique filename ..... mktemp(3C)
/gmtime, asctime, cftime, ascftime, mktime, tzset: convert date and time/ ..... ctime(3C)
mkvtoc: populate VTOC on disks ..... mkvtoc(1M)
m_lock, m_unlock: lock, unlock locks ..... m_lock(3PPS)
processs address space mmap: map an open file into the ..... mmap(2SEQ)
section m_single, m_multi: mark single-process code ..... m_single(3PPS)
m_next: increment global counter ..... m_next(3PPS)
mnttab: mounted filesystem table ..... mnttab(4)
configuration file mod_config: format of a module ..... mod_config(4)
chmod: change mode ..... chmod(1)
umask: set file-creation mode mask ..... umask(1)
chmod: change mode of file ..... chmod(2)
cpset: copy file and set mode, user, and group IDs ..... cpset(1)
getty: set terminal type, modes, speed, and line discipline ..... getty(1M)
uugetty: set terminal type, modes, speed, and line discipline ..... uugetty(1M)
frexp, ldexp, modf: manipulate parts of floating-point/ ..... frexp(3C)
utime: set file access and modification times ..... utime(2)
touch: update access and modification times of a file ..... touch(1)
astgen: program for generating/ modifying ASSIST menus or command/ ..... astgen(1)
printer canonical processing module ..... lpmod(7)
SSM: System Services Module ..... ssm(7)
Interface cooperating STREAMS module ..... timod(7)
read/write interface STREAMS module ..... tirdwr(7)
mod_config: format of a module configuration file ..... mod_config(4)
/ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily/ ..... acctsh(1M)
helplog: monitor help actions ..... helplog(4)
monitor: monitor system activity ..... monitor(1M)
monitor: prepare execution profile ..... monitor(3C)
monitor: monitor system activity ..... monitor(1M)
a time more: view a file one full screen at ..... more(1)
motd: message-of-the-day file ..... motd(4)
mount: mount a filesystem ..... mount(2)
mount, umount: mount and unmount filesystems ..... mount(1M)
mount: mount a filesystem ..... mount(2)
setmnt: establish mount table ..... setmnt(1M)
filesystems mount, umount: mount and unmount ..... mount(1M)

mountall, umountall: mount, unmount multiple filesystems.....mountall(1M)
multiple filesystems mountall, umountall: mount, unmount.....mountall(1M)
mnttab: mounted filesystem table.....mnttab(4)
mvdir: move a directory.....mvdir(1M)
cp, ln, mv: copy, link or move files.....cp(1)
lseek: move read/write file pointer.....lseek(2)
start/stop the LP print service and move requests.....lpsched(1M)
and resume child process execution m_park_procs, m_rele_procs: susp/....m_park_procs(3PPS)
ptrace, mptrace: process trace facility.....ptrace(2)
drand48, erand48, lrand48, nrand48, mrand48, jrand48, srand48, seed48,/.....drand48(3C)
child process/ m_park_procs, m_rele_procs: suspend and resume....m_park_procs(3PPS)
processes m_set_procs: set number of child.....m_set_procs(3PPS)
msgctl: message control operations.....msgctl(2)
msgget: get message queue.....msgget(2)
msgop: message operations.....msgop(2)
single-process code section m_single, m_multi: mark.....m_single(3PPS)
m_sync: check in at barrier.....m_sync(3PPS)
program mt: magnetic tape manipulating.....mt(1)
st: Systech MTI-800/1600/1650 terminal multiplexor driver.....st(7)
mtio: UNIX magnetic tape interface.....mtio(7)
mbad: MULITBUS adapter.....mbad(7)
xt: Xylogics 472 MULITBUS magnetic tape interface.....xt(7)
kMBmem, kMWmem: MULITBUS memory.....kmmem(7)
mountall, umountall: mount, unmount multiple filesystems.....mountall(1M)
poll: STREAMS input/output multiplexing.....poll(2)
Systech MTI-800/1600/1650 terminal multiplexor driver.....st(7)
xs: Xylogics 781 terminal multiplexor driver.....xs(7)
tmp_ctl: multi-processor control.....tmp_ctl(2SEQ)
rc2: run commands performed for multi-user environment.....rc2(1M)
m_lock, m_unlock: lock, unlock locks.....m_lock(3PPS)
address space munmap: unmap part of the processs.....munmap(2SEQ)
cp, ln, mv: copy, link or move files.....cp(1)
mvdir: move a directory.....mvdir(1M)
devnm: device name.....devnm(1M)
getenv: return value for environment name.....getenv(3C)
getgeombyname: get disk geometry by name.....getgeombyname(3SEQ)
getlogin: get login name.....getlogin(3C)
logname: get login name.....logname(1)
pwd: working directory name.....pwd(1)
tmpnam, tempnam: create a name for a temporary file.....tmpnam(3S)
table/ ldgetname: retrieve symbol name for common object file symbol.....ldgetname(3X)
getpw: get name from IUD.....getpw(3C)
nlist: get entries from name list.....nlist(3C)
rename: change the name of a file.....rename(2)
ttyname, isatty: find name of a terminal.....ttyname(3C)
uname: get name of current operating system.....uname(2)
uname: print name of current system.....uname(1)
tty: get the name of the terminal.....tty(1)
cuserid: get character login name of the user.....cuserid(3S)
logname: return login name of user.....logname(3X)
/ldnshread: read an indexed named section header of a common object/.....ldshread(3X)
/ldnsseek: seek to an indexed named section of a common object file.....ldsseek(3X)

between major/minor numbers and tty names.....dev_to_ttyname(3C)
id: print user and group IDs and names.....id(1M)
term: conventional names for terminals.....term(5)
isnanf: test for floating-point NaN (Not-A-Number).....isnan(3C)
processing language nawk: pattern scanning and .....nawk(1)
and indirect blocks nblocks: calculate number of blocks .....nblocks(3SEQ)
of filesystems processed by fsck and ncheck.....checklist(4)
i-numbers ncheck: generate pathnames from .....ncheck(1M)
uucp: networking support files.....uucp(4)
file newform: change the format of a text .....newform(1)
newfs: construct a new file system .....newfs(1M)
newgrp: log in to a new group.....newgrp(1M)
news: print news items.....news(1)
news: print news items.....news(1)
getmsg: get next message off a stream.....getmsg(2)
nice: change priority of a process.....nice(2)
nice: run a command at low priority .....nice(1)
nl: line numbering filter .....nl(1)
nlist: get entries from name list .....nlist(3C)
object file nm: print name list of.....see *C and Lang Tools Binder*
command with priority aging, PFF,/ noage, nopff, noswap, onproc: run a .....noage(1M)
hangups and quits nohup: run a command immune to .....nohup(1)
setjmp, longjmp: non-local goto.....setjmp(3C)
sigsetjmp, siglongjmp: non-local jumps.....sigsetjmp(3C)
with priority aging, PFF,/ noage, nopff, noswap, onproc: run a command.....noage(1M)
priority aging, PFF,/ noage, nopff, noswap, onproc: run a command with .....noage(1M)
test for floating-point NaN (Not-A-Number).....isnan(3C)
seed48,/ drand48, erand48, lrand48, nrand48, mrand48, jrand48, srand48, ...drand48(3C)
deroff: remove nroff, tbl, and eqn constructs.....deroff(1)
vax: get processor/ machid: i386, ns32000, pdp11, u3b, u3b2, u3b5,.....machid(1)
null: the null file.....null(7)
null: the null file .....null(7)
/ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct,/ .....acctsh(1M)
obtain the prime factors of a number.....factor(1)
getfrm: returns the current frame number.....getfrm(1F)
test for floating-point NaN (Not-A- Number) .....isnan(3C)
convert string to double-precision number.....strtod(3C)
linenum: line number entries in a common object file.....linenum(4)
/ldlinit, ldlitem: manipulate line number entries of a common object/.....ldlread(3X)
ldlseek, ldnlseek: seek to line number entries of a section of a/ .....ldlseek(3X)
rand, srand: simple random- number generator.....rand(3C)
nblocks: calculate number of blocks and indirect blocks.....nblocks(3SEQ)
m_get_numprocs: return number of child processes .....m_get_numprocs(3PPS)
m_set_procs: set number of child processes.....m_set_procs(3PPS)
cpus_online: returns the number of CPUs on-line .....cpus_online(3PPS)
df: report number of free disk blocks and inodes.....df(1M)
fcvt, gcvt: convert floating-point number to string .....ecvt(3C)
nl: line numbering filter .....nl(1)
uniformly distributed pseudorandom numbers.....drand48(3C)
manipulate parts of floating-point numbers .....frexp(3C)
install random inode generation numbers .....fsirand(1M)
to system calls and error numbers .....intro(2)

ncheck: generate pathnames from i- numbers.....ncheck(1M)
/convert between major/minor numbers and tty names.....dev_to_ttyname(3C)
cprs: compress an object file.....see *C and Lang Tools Binder*
strstr: find first occurrence of character.....strstr(3C)
od: octal dump.....od(1)
od: octal dump.....od(1)
offline: bring processors online and offline.....online(1M)
online, processor off line offline.....online(1M)
offline: bring processors online and offline.....online(1M)
online, offline: bring processors offline_all: take all but one active.....offline_all(3SEQ)
online and offline online and offline.....online(1M)
aging, PFF,/ noage, nopff, noswap, online, offline: bring processors.....online(1M)
ldopen, ldaopen: onproc: run a command with priority.....noage(1M)
fopen, freopen, fdopen: open a common object file for reading.....ldopen(3X)
clone: open a stream.....fopen(3S)
dup: duplicate an open any minor device on a STREAMS driver.....clone(7)
dup2: duplicate an open file descriptor.....dup(2)
mmap: map an open file descriptor.....dup2(2)
open: open file into the processs address space.....mmap(2SEQ)
open for reading or writing.....open(2)
open: open for reading or writing.....open(2)
rewinddir, closedir:/ directory: opendir, readdir, telldir, seekdir,.....directory(3X)
a standalone program or bring up the operating system.....boot(8)
run commands performed to stop the operating system.....rc0(1M)
savecore: save a core dump of the operating system.....savecore(1M)
uname: get name of current operating system.....uname(2)
locate: use keywords to identify an operating system command.....locate(1)
assist: assistance using operating system commands.....assist(1)
starter: information about the operating system for beginning users.....starter(1)
help: operating system Help Facility.....help(1)
glossary: definitions of common operating system terms and symbols.....glossary(1)
join: relational database operator.....join(1)
curses: terminal screen handling and optimization package.....curses(3X)
rmvopt: remove a menu option.....rmvopt(1M)
getopt: get option letter from argument vector.....getopt(3C)
addcmdopt: add a command option to a menu.....addcmdopt(1M)
addformopt: add a form/SELECT window option to a menu.....addformopt(1M)
addmenuopt: add a menu option to a menu.....addmenuopt(1M)
object file ldohseek: seek to the optional file header of a common.....ldohseek(3X)
fcntl: file control options.....fcntl(5)
getopt: parse command options.....getopt(1)
getopts, getoptcvt: parse command options.....getopts(1)
stty: set the options for a terminal.....stty(1)
t_optmgmt: manage options for a transport endpoint.....t_optmgmt(3N)
make a directory, or a special or ordinary file.....mknod(2)
architecture information ott: files that hold object.....ott(4)
common assembler and link editor output.....a.out(4)
echo: put string on virtual output.....echo(1F)
fread, fwrite: binary input/ output.....fread(3S)
fprintf, sprintf: print formatted output.....printf(3S)
poll: STREAMS input/ output multiplexing.....poll(2)
/vfprintf, vsprintf: print formatted output of a varargs argument list.....vprintf(3S)
stdio: standard buffered input/ output package.....stdio(3S)

setpgrp: run command in its own process group.....setpgrp(1)
chown: change owner and group of a file.....chown(2)
chown, chgrp: change owner or group.....chown(1)
expand files pack, pcat, unpack: compress and.....pack(1)
makewhatis: generate man page databases.....makealiases(1M)
getpagesize: get system page size.....getpagesize(2SEQ)
4014: paginator for the Tektronix 4014 terminal.....4014(1)
m_fork: execute a subprogram in parallel.....m_fork(3PPS)
lp: Systech parallel line printer interface.....lp(7)
wp: SSM parallel line printer interface.....wp(7)
lpmod: parallel printer canonical processing module.....lpmod(7)
intro: introduction to Parallel Programming Library.....intro(3PPS)
and change virtual memory tuning parameters.....vm_ctl(2SEQ)
tune system-wide virtual-memory parameters.....vmtune(1M)
pmap: display or alter parameters of virtual-to-physical mapping/.....pmap(1M)
get process, process group, and parent process IDs.....getpid(2)
getopt: parse command options.....getopt(1)
getopts, getoptcvt: parse command options.....getopts(1)
cmpt: compatibility partitioning driver.....cmpt(7)
vtoc: VTOC partitioning driver.....vtoc(7)
frexp, ldexp, modf: manipulate parts of floating-point numbers.....frexp(3C)
passmgmt: password files management.....passmgmt(1M)
with information from /etc/ passwd.....pwconv(1M)
password attributes passwd: change login password and.....passwd(1)
passwd: password file.....passwd(4)
getpass: read a password.....getpass(3C)
crypt: password and file encryption functions.....crypt(3X)
passwd: change login password and password attributes.....passwd(1)
passwd: change login password and password attributes.....passwd(1)
pwdhd: daemon to oversee password database creation.....pwdhd(1M)
pwfix: remove incorrectly formatted password entries.....pwfix(1M)
passwd: password file.....passwd(4)
shadow: restricted password file.....shadow(4)
setpwent, endpwent, fgetpwent: get password file entry.....getpwent(3C)
lckpwdf, ulckpwdf: get shadow password file entry.....getspent(3X)
putpwent: write password file entry.....putpwent(3C)
putspent: write shadow password file entry.....putspent(3X)
converts from a two to a one password file scheme.....pwunconv(1M)
passmgmt: password files management.....passmgmt(1M)
mkpwdbm: make a database for fast password lookup.....mkpwdbm(1M)
pwck, grpck: password/group file checkers.....pwck(1M)
files or subsequent lines of one/ paste: merge same lines of several.....paste(1)
bp: binary patcher.....bp(1M)
pathalias: alias file for FACE.....alias(4)
configurable pathname variables pathconf, fpathconf: get.....pathconf(2)
criteria for filename pathconv: search Interpreter.....pathconv(1F)
getcwd: get pathname of current working directory.....getcwd(3C)
fpathconf: get configurable pathname variables.....pathconf(2)
dirname: deliver portions of pathnames.....basename(1)
ncheck: generate pathnames from i-numbers.....ncheck(1M)
sweepon, sweepoff: enable/disable patrol seeks on DCC disks.....sweepon(1M)
grep: search a file for a pattern.....grep(1)

awk: pattern scanning and processing language ..... awk(1)
nawk: pattern scanning and processing language ..... nawk(1)
egrep: search a file for a pattern using full regular/..... egrep(1)
regex: match patterns against a string ..... regex(1F)
pause: suspend process until signal..... pause(2)
pack, pcat, unpack: compress and expand files ..... pack(1)
popen, pclose: initiate pipe to/from a process..... popen(3S)
pdbx, dbx: parallel/..... see *C and Lang Tools Binder*
machid: i386, ns32000, pdp11, u3b, u3b2, u3b5, vax: get processor/..... machid(1)
sigpending: examine pending signals ..... sigpending(2)
check the uucp directories and permissions file..... uucheck(1M)
acct: per-process accounting file format..... acct(4)
acctcms: command summary from per-process accounting records ..... acctcms(1M)
sys_nerr: system error messages perror, errno, sys_errlist, ... perror(3C)
run a command with priority aging, PFF, swapping, or processor/..... noage(1M)
pg: file perusal filter for CRTs..... pg(1)
virtual address space pmap: map physical addresses into processs ..... pmap(7)
or alter parameters of virtual-to- physical mapping devices..... pmap(1M)
pipe: create an interprocess channel..... pipe(2)
sp: STREAMS pipe device ..... sp(7)
tee: pipe fitting..... tee(1)
popen, pclose: initiate pipe to/from a process..... popen(3S)
in memory plock: lock process, text, or data ..... plock(2)
plot: graphics interface..... plot(4)
plot: graphics interface subroutines..... plot(3X)
virtual-to-physical mapping devices pmap: display or alter parameters of..... pmap(1M)
processs virtual address space pmap: map physical addresses into ..... pmap(7)
/fpsetsticky: IEEE floating- point environment control..... fpgetround(3C)
isnand, isnanf: test for floating- point NaN (Not-A-Number)..... isnan(3C)
ecvt, fcvt, gcvt: convert floating- point number to string ..... ecvt(3C)
modf: manipulate parts of floating- point numbers ..... frexp(3C)
lseek: move read/write file pointer..... lseek(2)
rewind, ftell: reposition a file pointer in a stream..... fseek(3S)
multiplexing poll: STREAMS input/output ..... poll(2)
a process popen, pclose: initiate pipe to/from ..... popen(3S)
banner: make posters ..... banner(1)
power, square root/ exp, log, log10, pow, sqrt: exponential, logarithm, ... exp(3M)
pow, sqrt: exponential, logarithm, power, square root functions..... exp(3M)
pr: print files..... pr(1)
/dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct,/ ..... acctsh(1M)
/lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup,/..... acctsh(1M)
bc: arbitrary- precision arithmetic language..... bc(1)
atof: convert string to double- precision number..... strtod(3C)
.environ, .pref, .variables: FACE files..... env(4)
monitor: prepare execution profile..... monitor(3C)
cpp: the C language preprocessor ..... see *C and Lang Tools Binder*
factor: obtain the prime factors of a number ..... factor(1)
gps: graphical primitive string, format of graphical files..... gps(4)
types: primitive system data types..... types(5)
prs: print an SCCS file..... prs(1)
date: print and set the date..... date(1)
cal: print calendar ..... cal(1)

sum: print checksum and block count of a file.....sum(1)
sact: print current SCCS file editing activity.....sact(1)
cat: concatenate and print files.....cat(1)
pr: print files.....pr(1)
printf, fprintf, sprintf: print formatted output.....printf(3S)
vprintf, vfprintf, vsprintf: print formatted output of a varargs/.....vprintf(3S)
of the LP print service lpstat: print information about the status.....lpstat(1)
information about RCS files rlog: print log messages and other.....rlog(1)
object file nm: print name list of.....see *C and Lang Tools Binder*
uname: print name of current system.....uname(1)
news: print news items.....news(1)
infocmp: compare or print out terminfo descriptions.....infocmp(1M)
acctcom: search and print process accounting files.....acctcom(1)
of object files size: print section sizes.....see *C and Lang Tools Binder*
send/cancel requests to the LP print service.....lp(1)
lpadmin: configure the LP print service.....lpadmin(1M)
administer filters used with the LP print service.....lpfilter(1M)
administer forms used with the LP print service.....lpforms(1M)
about the status of the LP print service.....lpstat(1)
/lpshut, lpmove: start/stop the LP print service and move requests.....lpsched(1M)
strace: print STREAMS trace messages.....strace(1M)
prtvtoc: print the VTOC of a block device.....prtvtoc(1M)
prtvtoc: print the VTOC of a standalone disk device.....prtvtoc(8)
id: print user and group IDs and names.....id(1M)
lpmod: parallel printer canonical processing module.....lpmod(7)
lp: Systech parallel line printer interface.....lp(7)
wp: SSM parallel line printer interface.....wp(7)
lptypes: table of printer types supported by the menu system.....lptypes(4)
enable, disable: enable/disable LP printers.....enable(1)
formatted output printf, fprintf, sprintf: print.....printf(3S)
lpusers: set printing queue priorities.....lpusers(1M)
lpusers: set printing queue priorities.....lpusers(1M)
nice: run a command at low priority.....nice(1)
/noswap, onproc: run a command with priority aging, PFF, swapping, or/.....noage(1M)
nice: change priority of a process.....nice(2)
brk, sbrk: change private data segment size.....brk(3PPS)
process attributes proc_ctl: control over various.....proc_ctl(2SEQ)
codestroy: communicate to a process.....coproc(1F)
exit, _exit: terminate process.....exit(2)
fork: create a new process.....fork(2)
inittab: script for the init process.....inittab(4)
kill: terminate a process.....kill(1)
nice: change priority of a process.....nice(2)
pclose: initiate pipe to/from a process.....popen(3S)
wait: await completion of process.....wait(1)
acct: enable or disable process accounting.....acct(2)
acctprc1, acctprc2: process accounting.....acctprc(1M)
acct: per- process accounting file format.....acct(4)
acctcom: search and print process accounting files.....acctcom(1)
acctcms: command summary from per- process accounting records.....acctcms(1M)
alarm: set a process alarm clock.....alarm(2)
times: get process and child process times.....times(2)

proc_ctl: control over various process attributes.....proc_ctl(2SEQ)
m_single, m_multi: mark single- process code section.....m_single(3PPS)
ipcs: report inter- process communication facilities status.....ipcs(1)
init, telinit: process control initialization.....init(1M)
timex: time a command; report process data and system activity.....timex(1)
suspend and resume child process execution.....m_park_procs(3PPS)
setpgrp: run command in its own process group.....setpgrp(1)
/getpgrp, getppid: get process, process group, and parent process IDs.....getpid(2)
setpgrp: set process group ID.....setpgrp(2)
setsid: create session and set process group ID.....setsid(2)
tcgetpgrp: get foreground process group ID.....tcgetpgrp(2)
tcsetpgrp: set foreground process group ID.....tcsetpgrp(2)
m_get_myid: return process identification.....m_get_myid(3PPS)
process, process group, and parent process IDs.....getpid(2)
vfork: spawn new process in a virtual memory efficient way.....vfork(2)
kill: send a signal to a process or a group of processes.....kill(2)
getpid, getpgrp, getppid: get process, process group, and parent/.....getpid(2)
ps: report process status.....ps(1)
plock: lock process, text, or data in memory.....plock(2)
times: get process and child process times.....times(2)
tmp_affinity: bind a process to a processor.....tmp_affinity(2SEQ)
wait, waitpid: wait for a child process to stop or terminate.....wait(2)
ptrace, mptrace: process trace facility.....ptrace(2)
pause: suspend process until signal.....pause(2)
vsig: synchronize a co- process with the controlling FMLI object.....vsig(1F)
checklist: list of filesystems processed by fsck and ncheck.....checklist(4)
a signal to a process or a group of processes.....kill(2)
killall: kill all active processes.....killall(1M)
return number of child processes.....m_get_numprocs(3PPS)
m_kill_procs: kill child processes.....m_kill_procs(3PPS)
m_set_procs: set number of child processes.....m_set_procs(3PPS)
fuser: identify processes using a file or file structure.....fuser(1M)
m4: macro processor.....m4(1)
menu: menu processor.....menu(1M)
tmp_affinity: bind a process to a processor.....tmp_affinity(2SEQ)
tmp_ctl: multi- processor control.....tmp_ctl(2SEQ)
priority aging, PFF, swapping, or processor migration disabled.....noage(1M)
offline_all: take all but one active processor off line.....offline_all(3SEQ)
pdp11, u3b, u3b2, u3b5, vax: get processor type truth value.....machid(1)
online, offline: bring processors online and offline.....online(1M)
mmap: map an open file into the processs address space.....mmap(2SEQ)
munmap: unmap part of the processs address space.....munmap(2SEQ)
pmap: map physical addresses into processs virtual address space.....pmap(7)
product: installed products file.....product(4)
prof: display profile data.....prof(1)
prof: profile within a function.....prof(5)
profil: execution time profile.....profil(2)
monitor: prepare execution profile.....monitor(3C)
profil: execution time profile.....profil(2)
prof: display profile data.....prof(1)
at login time profile: setting up an environment.....profile(4)
prof: profile within a function.....prof(5)

end, etext, edata: last locations in program ..... end(3C)
mt: magnetic tape manipulating program ..... mt(1)
prompt: prompt program ..... prompt(1M)
sdiff: side-by-side difference program ..... sdiff(1)
STREAMS error logger cleanup program ..... strclean(1M)
units: conversion program ..... units(1)
for the uucp file transport program ..... uusched(1M)
assert: verify program assertion ..... assert(3X)
ASSIST menus or command/ astgen: program for generating/modifying ..... astgen(1)
uucico: file transport program for the uucp system ..... uucico(1M)
installboot: installs a boot program on a disk ..... installboot(1M)
boot: run a standalone program or bring up the operating system ..... boot(8)
a standard/restricted command/ programming language ..... ksh(1)
the standard/restricted command programming language ..... sh(1)
intro: introduction to Parallel Programming Library ..... intro(3PPS)
maintenance commands and application programs ..... intro(1)
update, and regenerate groups of programs ..... make(1)
lex: generate programs for simple lexical tasks ..... lex(1)
setlocale: set programs location ..... setlocale(3C)
prompt: prompt program ..... prompt(1M)
prompt: prompt program ..... prompt(1M)
proto: prototype job file for at ..... proto(4)
t_getinfo: get protocol-specific service information ..... t_getinfo(3N)
proto: prototype job file for at ..... proto(4)
prs: print an SCCS file ..... prs(1)
/monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup/ ..... acctsh(1M)
device prtvtoc: print the VTOC of a block ..... prtvtoc(1M)
standalone disk device prtvtoc: print the VTOC of a ..... prtvtoc(8)
ps: report process status ..... ps(1)
streampty: STREAMS pseudo terminal driver ..... streampty(7)
getpseudotty: get pseudo terminals ..... getpseudotty(3SEQ)
cn: pseudoconsole driver ..... cn(7)
generate uniformly distributed pseudorandom numbers ..... drand48(3C)
time, ptime: time a command ..... time(1)
facility ptrace, mptrace: process trace ..... ptrace(2)
uuto, uupick: public UNIX-to-UNIX system file copy ..... uuto(1C)
ungetc: push character back into input stream ..... ungetc(3S)
puts, fputs: put a string on a stream ..... puts(3S)
putc, putchar, fputc, putw: put character or word on a stream ..... putc(3S)
getdents: read directory entries and put in a file system independent/ ..... getdents(2)
echo: put string on virtual output ..... echo(1F)
character or word on a stream putc, putchar, fputc, putw: put ..... putc(3S)
or word on a stream putc, putchar, fputc, putw: put character ..... putc(3S)
environment putenv: change or add value to ..... putenv(3C)
putmsg: send a message on a stream ..... putmsg(2)
putpwent: write password file entry ..... putpwent(3C)
stream puts, fputs: put a string on a ..... puts(3S)
message: puts its arguments on message line ..... message(1F)
entry putspent: write shadow password file ..... putspent(3X)
getut: getutent, getutid, getutline, pututline, setutent, endutent/ ..... getut(3C)
putc, putchar, fputc, putw: put character or word on a stream ..... putc(3S)
checkers pwck, grpck: password/group file ..... pwck(1M)

/etc/shadow with information from/ pwconv: installs and updates ..... pwconv(1M)
pwd: working directory name ..... pwd(1)
database creation pwdhd: daemon to oversee password ..... pwdhd(1M)
password entries pwfix: remove incorrectly formatted ..... pwfix(1M)
one password file scheme pwunconv: converts from a two to a ..... pwunconv(1M)
qsort: quicker sort ..... qsort(3C)
tput: initialize a terminal or query terminfo database ..... tput(1)
route user message to menu queue ..... menumsg(1M)
msgget: get message queue ..... msgget(2)
lpusers: set printing queue priorities ..... lpusers(1M)
ipcrm: remove a message queue, semaphore set, or shared memory ID ..... ipcrm(1)
quotacheck: filesystem quota consistency checker ..... quotacheck(1M)
quota: display disk usage and limits ..... quota(1)
consistency checker quotacheck: filesystem quota ..... quotacheck(1M)
quotactl: manipulate disk quotas ..... quotactl(2SEQ)
quotaon, quotaoff: turn filesystem quotas on and off ..... quotaon(1M)
quotas on and off quotaon, quotaoff: turn filesystem ..... quotaon(1M)
edquota: edit user quotas ..... edquota(1M)
quotactl: manipulate disk quotas ..... quotactl(2SEQ)
repquota: summarize quotas for a filesystem ..... repquota(1M)
quotaon, quotaoff: turn filesystem quotas on and off ..... quotaon(1M)
generator rand, srand: simple random-number ..... rand(3C)
fsirand: install random inode generation numbers ..... fsirand(1M)
rand, srand: simple random-number generator ..... rand(3C)
fsplit: split f77, ratfor, or efl files ..... fsplit(1)
ratfor: rational Fortran dialect ..... ratfor(1)
ratfor: rational Fortran dialect ..... ratfor(1)
scripts rc: reboot and shutdown command ..... rc(4)
the operating system rc0: run commands performed to stop ..... rc0(1M)
multi-user environment rc2: run commands performed for ..... rc2(1M)
rcs: change RCS file attributes ..... rcs(1)
rcsintro: introduction to RCS commands ..... rcsintro(1)
rcsfile: format of RCS file ..... rcsfile(4)
rcs: change RCS file attributes ..... rcs(1)
sccstorcs: build RCS file from SCCS file ..... sccstorcs(1)
messages and other information about RCS files ..... rlog(1)
ci: check in RCS revisions ..... ci(1)
co: check out RCS revisions ..... co(1)
rcsdiff: compare RCS revisions ..... rcsdiff(1)
rcsmerge: merge RCS revisions ..... rcsmerge(1)
rcsdiff: compare RCS revisions ..... rcsdiff(1)
rcsfile: format of RCS file ..... rcsfile(4)
commands rcsintro: introduction to RCS ..... rcsintro(1)
rcsmerge: merge RCS revisions ..... rcsmerge(1)
getpass: read a password ..... getpass(3C)
of a common object file ldtbread: read an indexed symbol table entry ..... ldtbread(3X)
header of a/ ldshread, ldnshread: read an indexednamed section ..... ldshread(3X)
structure read_constab: read constab entries into a ..... read_constab(3SEQ)
file system independent/ getdents: read directory entries and put in a ..... getdents(2)
read: read from file ..... read(2)
mail, rmail: send mail to users or read mail ..... mail(1)
line: read one line ..... line(1)

read: read from file ..... read(2)
of an archive file ldahread: read the archive header of a member ..... ldahread(3X)
object file ldfhread: read the file header of a common ..... ldfhread(3X)
readlink: read value of a symbolic link ..... readlink(2SEQ)
readv: read vector from file ..... readv(2SEQ)
into a structure read_constab: read constab entries ..... read_constab(3SEQ)
directory: opendir, readdir, telldir, seekdir, rewinddir/ ..... directory(3X)
gets longest line readfile, longline: reads file and ..... readfile(1F)
open a common object file for reading ..... ldopen(3X)
open: open for reading or writing ..... open(2)
link readlink: read value of a symbolic ..... readlink(2SEQ)
readfile, longline: reads file and gets longest line ..... readfile(1F)
readv: read vector from file ..... readv(2SEQ)
lseek: move read/write file pointer ..... lseek(2)
tirdwr: Transport Interface read/write interface STREAMS module ..... tirdwr(7)
/get real user, effective user, real group, and effective group IDs ..... getuid(2)
/geteuid, getgid, getegid: get real user, effective user, real/ ..... getuid(2)
malloc, free, realloc, calloc: main memory allocator ..... malloc(3C)
fast main memory/ malloc, free, realloc, calloc, mallopt, mallinfo: ..... malloc(3X)
rc: reboot and shutdown command scripts ..... rc(4)
t_rcvudata: receive a data unit ..... t_rcvudata(3N)
t_rcvuderr: receive a unit data error indication ..... t_rcvuderr(3N)
over a connection t_rcv: receive data or expedited data sent ..... t_rcv(3N)
connect request t_rcvconnect: receive the confirmation from a ..... t_rcvconnect(3N)
lockf: record locking on files ..... lockf(3C)
ed, red: text editor ..... ed(1)
setcolor: redefine or create a color ..... setcolor(1F)
dltsectbl: refresh security table ..... dltsectbl(1M)
regular expression regcmp, regex: compile and execute ..... regcmp(3X)
regcmp: regular expression compile ..... regcmp(1)
make: maintain, update, and regenerate groups of programs ..... make(1)
expression regcmp, regex: compile and execute regular ..... regcmp(3X)
string regex: match patterns against a ..... regex(1F)
and match routines regexp: regular expression compile ..... regexp(5)
regcmp, regex: compile and execute regular expression ..... regcmp(3X)
regcmp: regular expression compile ..... regcmp(1)
routines regexp: regular expression compile and match ..... regexp(5)
a file for a pattern using full regular expressions ..... egrep(1)
reinit: run an initialization file ..... reinit(1F)
accept, reject: allow or prevent LP requests ..... accept(1M)
comm: select or reject lines common to two sorted files ..... comm(1)
join: relational database operator ..... join(1)
t_sndrel: initiate an orderly release ..... t_sndrel(3N)
acknowledge receipt of an orderly release indication ..... t_rcvrel(3N)
common object file reloc: relocation information for a ..... reloc(4)
ldrseek, ldnrseek: seek to relocation entries of a section of a common/ ..... ldrseek(3X)
reloc: relocation information for a common object file ..... reloc(4)
/ceil, fmod, fabs: floor, ceiling, remainder, absolute value functions ..... floor(3M)
calendar: reminder service ..... calendar(1)
uuxqt: execute remote command requests ..... uuxqt(1M)
Uutry: try to contact remote system with debugging on ..... uutry(1M)
ct: spawn getty to a remote terminal ..... ct(1C)

rmdel: remove a delta from an SCCS file.....rmdel(1)
rmdir: remove a directory.....rmdir(2)
rmvopt: remove a menu option.....rmvopt(1M)
set, or shared memory ID ipcrm: remove a message queue, semaphore.....ipcrm(1)
unlink: remove directory entry.....unlink(2)
rm, rmdir: remove files or directories.....rm(1)
password entries pwfix: remove incorrectly formatted.....pwfix(1M)
deroff: remove nroff, tbl, and eqn constructs.....deroff(1)
rename: change the name of a file.....rename(2)
consistency check and interactive repair.....fsck(1M)
uniq: report repeated lines in a file.....uniq(1)
clock: report CPU time used.....clock(3C)
fsstat: report filesystem status.....fsstat(1M)
facilities status ipcs: report inter-process communication.....ipcs(1)
df: report number of free disk blocks and inodes.....df(1M)
mailbug: submit a bug report or enhancement request.....mailbug(1)
sar: sa1, sa2, sadc: system activity report package.....sar(1M)
timex: time a command; report process data and system activity.....timex(1)
ps: report process status.....ps(1)
uniq: report repeated lines in a file.....uniq(1)
fseek, rewind, ftell: reposition a file pointer in a stream.....fseek(3S)
filesystem repquota: summarize quotas for a.....repquota(1M)
submit a bug report or enhancement request.....mailbug(1)
t_accept: accept a connect request.....t_accept(3N)
t_listen: listen for a connect request.....t_listen(3N)
the confirmation from a connect request.....t_rcvconnect(3N)
send user-initiated disconnect request.....t_snddis(3N)
accept, reject: allow or prevent LP requests.....accept(1M)
the LP print service and move requests.....lpsched(1M)
uuxqt: execute remote command requests.....uuxqt(1M)
lp, cancel: send/cancel requests to the LP print service.....lp(1)
to its default values reset: reset the current form field.....reset(1F)
default values reset: reset the current form field to its.....reset(1F)
get_process_stats: get resource utilization info.....get_process_stats(2SEQ)
restore: incremental filesystem restore.....restore(1M)
restore restore: incremental filesystem.....restore(1M)
/ksh, rksh – KornShell, a standard/ restricted command and programming/.....ksh(1)
sh, rsh: shell, the standard/ restricted command programming/.....sh(1)
shadow: restricted password file.....shadow(4)
/m_rele_procs: suspend and resume child process execution.....m_park_procs(3PPS)
usage examples usage: retrieve a command description and.....usage(1)
t_rcvdis: retrieve information from disconnect.....t_rcvdis(3N)
object file symbol table/ ldgetname: retrieve symbol name for common.....ldgetname(3X)
abs: return integer absolute value.....abs(3C)
logname: return login name of user.....logname(3X)
m_get_numprocs: return number of child procs.....m_get_numprocs(3PPS)
m_get_myid: return process identification.....m_get_myid(3PPS)
getenv: return value for environment name.....getenv(3C)
stat: data returned by stat system call.....stat(5)
marked menu items getitems: returns a list of the currently.....getitems(1F)
getfrm: returns the current frame number.....getfrm(1F)
cpus_online: returns the number of CPUs on-line.....cpus_online(3PPS)

col: filter reverse line-feeds.....col(1)
ci: check in RCS revisions.....ci(1)
co: check out RCS revisions.....co(1)
rcsdiff: compare RCS revisions.....rcsdiff(1)
rcsmerge: merge RCS revisions.....rcsmerge(1)
pointer in a stream fseek, rewind, ftell: reposition a file.....fseek(3S)
/opendir, readdir, telldir, seekdir, rewinddir, closedir: directory/.....directory(3X)
creat: create a new file or rewrite an existing one.....creat(2)
restricted command/ ksh, rksh: KornShell, a standard/.....ksh(1)
information about RCS files rlog: print log messages and other.....rlog(1)
directories rm, rmdir: remove files or.....rm(1)
mail, rmail: send mail to users or read mail.....mail(1)
file rmdel: remove a delta from an SCCS.....rmdel(1)
rmdir: remove a directory.....rmdir(2)
rm, rmdir: remove files or directories.....rm(1)
rmvopt: remove a menu option.....rmvopt(1M)
chroot: change root directory.....chroot(2)
chroot: change root directory for a command.....chroot(1M)
logarithm, power, square root functions.....exp(3M)
queue menumsg: route user message to menu message.....menumsg(1M)
ldfcn: common object file access routines.....ldfcn(4)
regular expression compile and match routines.....regexp(5)
programming language sh, rsh: shell, standard/restricted command/.....sh(1)
nice: run a command at low priority.....nice(1)
nohup: run a command immune to hangups and quits.....nohup(1)
shell: run a command using shell.....shell(1F)
noage, nopff, noswap, onproc: run a command with priority aging, PFF/.....noage(1M)
the operating system boot: run a standalone program or bring up.....boot(8)
run: run an executable.....run(1F)
edc: run an externally described command.....edc(1M)
reinit: run an initialization file.....reinit(1F)
setpgrp: run command in its own process group.....setpgrp(1)
multi-user environment rc2: run commands performed for.....rc2(1M)
operating system rc0: run commands performed to stop the.....rc0(1M)
runacct: run daily accounting.....runacct(1M)
run: run an executable.....run(1F)
runacct: run daily accounting.....runacct(1M)
Administration SA: devices administered by System.....sa(7)
sar: sa1, sa2, sadc: system activity report package.....sar(1M)
sar: sa1, sa2, sadc: system activity report package.....sar(1M)
editing activity sact: print current SCCS file.....sact(1)
sar: sa1, sa2, sadc: system activity report package.....sar(1M)
report package sar: sa1, sa2, sadc: system activity.....sar(1M)
sar: system activity reporter.....sar(1)
savecore: save a core dump of the operating system.....savecore(1M)
operating system savecore: save a core dump of the.....savecore(1M)
brk, sbrk: change data segment space allocation.....brk(2)
brk, sbrk: change private data segment size.....brk(3PPS)
formatted input scanf, fscanf, sscanf: convert.....scanf(3S)
bfs: big file scanner.....bfs(1)
awk: pattern scanning and processing language.....awk(1)
nawk: pattern scanning and processing language.....nawk(1)

change the delta commentary of an SCCS delta ..... cdc(1)
comb: combine SCCS deltas ..... comb(1)
delta: make a delta (change) to an SCCS file ..... delta(1)
get: get a version of an SCCS file ..... get(1)
prs: print an SCCS file ..... prs(1)
rmdel: remove a delta from an SCCS file ..... rmdel(1)
sccsdiff: compare two versions of an SCCS file ..... sccsdiff(1)
sccsfile: format of SCCS file ..... sccsfile(4)
sccstorcs: build RCS file from SCCS file ..... sccstorcs(1)
unget: undo a previous get of an SCCS file ..... unget(1)
val: validate SCCS file ..... val(1)
sact: print current SCCS file editing activity ..... sact(1)
admin: create and administer SCCS files ..... admin(1)
what: identify SCCS files ..... what(1)
SCCS file sccsdiff: compare two versions of an ..... sccsdiff(1)
sccsfile: format of SCCS file ..... sccsfile(4)
file sccstorcs: build RCS file from SCCS ..... sccstorcs(1)
board SCED: SCSI/Ethernet/Diagnostics ..... sced(7)
ckbupscd: check filesystem backup schedule ..... ckbupscd(1M)
transport program uusched: the scheduler for the uucp file ..... uusched(1M)
unlock locks s_init_lock, s_lock, s_clock, s_unlock: initialize, lock, ... s_lock(3PPS)
object file scnhdr: section header for a common ..... scnhdr(4)
image file scr_dump: format of curses screen ..... scr_dump(4)
more: view a file one full screen at a time ..... more(1)
curses: terminal screen handling and optimization package ..... curses(3X)
scr_dump: format of curses screen image file ..... scr_dump(4)
vi: screen-oriented (visual) display editor based on ex ..... vi(1)
inittab: script for the init process ..... inittab(4)
rc: reboot and shutdown command scripts ..... rc(4)
tg: SCSI 1/2-inch magnetic tape interface ..... tg(7)
sd: SCSI disk controller ..... sd(7)
wd: SSM SCSI disk controller ..... wd(7)
CCSformat: standalone embedded SCSI disk formatter ..... ccsformat(8)
tm: SCSI tape controller ..... tm(7)
ts: SCSI tape controller ..... ts(7)
SCED: SCSI/Ethernet/Diagnostics board ..... sced(7)
getscsiinfo, getscsimatch: get scsiinfo structure ..... getscsiinfo(3SEQ)
sd: SCSI disk controller ..... sd(7)
program sdiff: side-by-side difference ..... sdiff(1)
fgrep: search a file for a character string ..... fgrep(1)
grep: search a file for a pattern ..... grep(1)
full regular expressions egrep: search a file for a pattern using ..... egrep(1)
bsearch: binary search a sorted table ..... bsearch(3C)
acctcom: search and print process accounting files ..... acctcom(1)
lsearch, lfind: linear search and update ..... lsearch(3C)
pathconv: search Interpreter criteria for filename ..... pathconv(1F)
hcreate, hdestroy: manage hash search tables ..... hsearch(3C)
tfind, tdelete, twalk: manage binary search trees ..... tsearch(3C)
manipulate the object file comment section ..... mcs(1)
m_multi: mark single-process code section ..... m_single(3PPS)
scnhdr: section header for a common object file ..... scnhdr(4)
/ldnshread: read an indexednamed section header of a common object/ ..... ldshread(3X)

/seek to line number entries of a section of a common object file ..... ldlseek(3X)
/seek to relocation entries of a section of a common object file ..... ldrseek(3X)
/ldnsseek: seek to an indexednamed section of a common object file ..... ldsseek(3X)
dltsectbl: refresh security table ..... dltsectbl(1M)
task: global security table ..... task(4)
tasktbl: create security table in shared memory ..... tasktbl(1M)
sed: stream editor ..... sed(1)
/nrand48, mrand48, jrand48, srand48, seed48, lcong48: generate uniformly/ ..... drand48(3C)
a common object/ ldsseek, ldnsseek: seek to an indexednamed section of ..... ldsseek(3X)
section of a/ ldlseek, ldnlseek: seek to line number entries of a ..... ldlseek(3X)
ldrseek, ldnrseek: seek to relocation entries of a section of a/ ..... ldrseek(3X)
a common object file ldohseek: seek to the optional file header of ..... ldohseek(3X)
object file ldtbseek: seek to the symbol table of a common ..... ldtbseek(3X)
/opendir, readdir, telldir, seekdir, rewinddir, closedir/ ..... directory(3X)
sweepoff: enable/disable patrol seeks on DCC disks ..... sweepon(1M)
shmget: get shared memory segment identifier ..... shmget(2)
brk, sbrk: change private data segment size ..... brk(3PPS)
shbrk, shsbrk: change shared data segment size ..... shbrk(3PPS)
brk, sbrk: change data segment space allocation ..... brk(2)
sorted files comm: select or reject lines common to two ..... comm(1)
greek: select terminal filter ..... greek(1)
addformopt: add a form/ SELECT window option to a menu ..... addformopt(1M)
semctl: semaphore control operations ..... semctl(2)
semop: semaphore operations ..... semop(2)
ipcrm: remove a message queue, semaphore set, or shared memory ID ..... ipcrm(1)
semget: get set of semaphores ..... semget(2)
semctl: semaphore control operations ..... semctl(2)
semget: get set of semaphores ..... semget(2)
semop: semaphore operations ..... semop(2)
t_sndudata: send a data unit ..... t_sndudata(3N)
putmsg: send a message on a stream ..... putmsg(2)
kill: send a signal to a process or a group of processes ..... kill(2)
tfnotify: send completion message ..... tfnotify(1M)
connection t_snd: send data or expedited data over a ..... t_snd(3N)
mail, rmail: send mail to users or read mail ..... mail(1)
t_snddis: send user-initiated disconnect request ..... t_snddis(3N)
lp, cancel: send/cancel requests to the LP print service ..... lp(1)
receive data or expedited data sent over a connection ..... t_rcv(3N)
intro: introduction to Sequent system calls ..... intro(2SEQ)
dis386: disassemble Series 386 instructions ..... dis386(3SEQ)
SSM: System Services Module ..... ssm(7)
setsid: create session and set process group ID ..... setsid(2)
ascii: map of ASCII character set ..... ascii(5)
alarm: set a process alarm clock ..... alarm(2)
umask: set and get file creation mask ..... umask(2)
stty, gtty: set and get terminal state (defunct) ..... stty(2)
in core or in files set, unset: set and unset environment variables ..... set(1F)
sigstack: set and/or get signal stack context ..... sigstack(2SEQ)
timezone: set default system time zone ..... timezone(4)
backup.info: backup set definitions ..... backup_info(4)
setdtablesize: set descriptor table size ..... setdtablesize(2SEQ)
env: set environment for command execution ..... env(1)

utime: set file access and modification times.....utime(2)
umask: set file-creation mode mask .....umask(1)
tcsetpgrp: set foreground process group ID.....tcsetpgrp(2)
setgroups: set group access list .....setgroups(2SEQ)
cpset: copy file and set mode, user, and group IDs.....cpset(1)
m_set_procs: set number of child processes.....m_set_procs(3PPS)
semget: get set of semaphores.....semget(2)
remove a message queue, semaphore set, or shared memory ID.....ipcrm(1)
lpusers: set printing queue priorities.....lpusers(1M)
setpgrp: set process group ID.....setpgrp(2)
setsid: create session and set process group ID.....setsid(2)
setlocale: set programs location .....setlocale(3C)
tabs: set tabs on a terminal.....tabs(1)
tcgetattr, tcsetattr: get/ set terminal state .....tcgetattr(2)
line discipline getty: set terminal type, modes, speed, and.....getty(1M)
line discipline uugetty: set terminal type, modes, speed, and.....uugetty(1M)
date: print and set the date .....date(1)
stty: set the options for a terminal .....stty(1)
stime: set time.....stime(2)
environment variables in core or in/ set, unset: set and unset .....set(1F)
setuid, setgid: set user and group IDs.....setuid(2)
ulimit: get and set user limits.....ulimit(2)
a stream setbuf, setvbuf: assign buffering to.....setbuf(3S)
/toascci, _tolower, _toupper, setchrclass: character handling.....ctype(3C)
setcolor: redefine or create a color .....setcolor(1F)
size setdtablesize: set descriptor table.....setdtablesize(2SEQ)
setuid, setgid: set user and group IDs.....setuid(2)
getgrent, getgrgid, getgrnam, setgrent, endgrent, fgetgrent: get group/.....getgrent(3C)
setgroups: set group access list.....setgroups(2SEQ)
setjmp, longjmp: non-local goto .....setjmp(3C)
crypt, setkey, encrypt: generate hashing encryption .....crypt(3C)
setlocale: set programs location .....setlocale(3C)
setmnt: establish mount table.....setmnt(1M)
/endmntent, getmntent, hasmntopt, setmntent: get file system/.....getmntent(3SEQ)
setpgid: manipulate signal sets.....setpgid(2)
process group setpgrp: run command in its own.....setpgrp(1)
setpgrp: set process group ID.....setpgrp(2)
getpwent, getpwuid, getpwnam, setpwent, endpwent, fgetpwent: get/.....getpwent(3C)
process group ID setsid: create session and set.....setsid(2)
getspent, getspnam, setspent, endspent, fgetspent, lckpwdf/.....getspent(3X)
profile: setting up an environment at login time .....profile(4)
gettydefs: speed and terminal settings used by getty.....gettydefs(4)
IDs setuid, setgid: set user and group.....setuid(2)
user setup: initialize system for first.....setup(1)
/getutid, getutline, pututline, setutent, endutent, utmpname/.....getut(3C)
setbuf, setvbuf: assign buffering to a stream .....setbuf(3S)
machine-independent fashion sputl, sgetl: access long integer data in a.....sputl(3X)
standard/restricted command/ sh, rsh: shell, the.....sh(1)
/fgetspent, lckpwdf, ulckpwdf: get shadow password file entry.....getspent(3X)
putspent: write shadow password file entry.....putspent(3X)
shadow: restricted password file.....shadow(4)
pwconv: installs and updates /etc/ shadow with information from/.....pwconv(1M)

shbrk, shsbrk: change ... shared data segment size ... shbrk(3PPS)
tasktbl: create security table in ... shared memory ... tasktbl(1M)
shmalloc, shrealloc, shfree: ... shared memory allocator ... shmalloc(3PPS)
shmctl: ... shared memory control operations ... shmctl(2)
a message queue, semaphore set, or ... shared memory ID ... ipcrm(1)
shmop, shmat, shmdt: ... shared memory operations ... shmop(2)
shmget: get ... shared memory segment identifier ... shmget(2)
segment size ... shbrk, shsbrk: change shared data ... shbrk(3PPS)
shell: run a command using ... shell ... shell(1F)
assign menu system as user login ... shell ... tfmenush(1M)
system: issue a ... shell command ... system(3S)
csh: a ... shell (command interpreter) with C-like syntax ... csh(1)
/shutacct, startup, turnacct: ... shell procedures for accounting ... acctsh(1M)
... shell: run a command using shell ... shell(1F)
shells: ... shell table ... shells(4)
command programming/ sh, rsh: ... shell, the standard/restricted ... sh(1)
... shells: shell table ... shells(4)
shmalloc, shrealloc, ... shfree: shared memory allocator ... shmalloc(3PPS)
memory allocator ... shmalloc, shrealloc, shfree: shared ... shmalloc(3PPS)
shmop, ... shmat, shmdt: shared memory operations ... shmop(2)
operations ... shmctl: shared memory control ... shmctl(2)
shmop, shmat, ... shmdt: shared memory operations ... shmop(2)
identifier ... shmget: get shared memory segment ... shmget(2)
operations ... shmop, shmat, shmdt: shared memory ... shmop(2)
the machine ... showcfg: show the configuration of ... showcfg(1M)
allocator shmalloc, ... shrealloc, shfree: shared memory ... shmalloc(3PPS)
shbrk, ... shsbrk: change shared data segment size ... shbrk(3PPS)
shutdown: ... shut down system, change system state ... shutdown(1M)
/nulladm, prctmp, prdaily, prtacct, ... shutacct, startup, turnacct: shell/ ... acctsh(1M)
rc: reboot and ... shutdown command scripts ... rc(4)
shutdown: system ... shutdown information file ... shutdown(4)
system state ... shutdown: shut down system, change ... shutdown(1M)
information file ... shutdown: system shutdown ... shutdown(4)
actions ... sigaction: examine and change signal ... sigaction(2)
manipulate/ sigemptyset, sigfillset, ... sigaddset, sigdelset, sigismember: ... sigemptyset(2)
... sigcontext: get signal context ... sigcontext(2SEQ)
sigemptyset, sigfillset, sigaddset, ... sigdelset, sigismember: manipulate/ ... sigemptyset(2)
sigdelset, sigismember: manipulate/ ... sigemptyset, sigfillset, sigaddset, ... sigemptyset(2)
sigismember/ sigemptyset, ... sigfillset, sigaddset, sigdelset, ... sigemptyset(2)
sigpause: signal management sigset, ... sighold, sigrelse, sigignore, ... sigset(2)
sigset, sighold, sigrelse, ... sigignore, sigpause: signal/ ... sigset(2)
/sigfillset, sigaddset, sigdelset, ... sigismember: manipulate signal sets ... sigemptyset(2)
sigsetjmp, ... siglongjmp: non-local jumps ... sigsetjmp(3C)
login: ... sign on ... login(1)
pause: suspend process until ... signal ... pause(2)
specify what to do upon receipt of a ... signal ... signal(2)
sigsuspend: wait for a ... signal ... sigsuspend(2)
sigaction: examine and change ... signal actions ... sigaction(2)
sigcontext: get ... signal context ... sigcontext(2SEQ)
sigrelse, sigignore, sigpause: ... signal management ... sigset(2)
setpgid: manipulate ... signal sets ... setpgid(2)
sigdelset, sigismember: manipulate ... signal sets ... sigemptyset(2)

receipt of a signal — signal: specify what to do upon ..... signal(2)
sigstack: set and/or get — signal stack context ..... sigstack(2SEQ)
kill: send a — signal to a process or a group of processes ..... kill(2)
sigpending: examine pending — signals ..... sigpending(2)
examine and change blocked — signals ..... sigprocmask(2)
ssignal, gsignal: software — signals ..... ssignal(3C)
sighold, sigrelse, sigignore, — sigpause: signal management ..... sigset(2)
— sigpending: examine pending signals ..... sigpending(2)
blocked signals — sigprocmask: examine and change ..... sigprocmask(2)
signal management sigset, sighold, — sigrelse, sigignore, sigpause: ..... sigset(2)
sigignore, sigpause: signal/ — sigset, sighold, sigrelse, ... sigset(2)
jumps — sigsetjmp, siglongjmp: non-local ..... sigsetjmp(3C)
stack context — sigstack: set and/or get signal ..... sigstack(2SEQ)
— sigsuspend: wait for a signal ..... sigsuspend(2)
atan2: trigonometric/ trig: — sin, cos, tan, asin, acos, atan, ... trig(3M)
m_single, m_multi: mark — single-process code section ..... m_single(3PPS)
functions — sinh, cosh, tanh: hyperbolic ..... sinh(3M)
initialize barrier, wait at barrier — s_init_barrier, s_wait_barrier: ..... s_wait_barrier(3PPS)
s_unlock: initialize, lock, unlock/ — s_init_lock, s_lock, s_clock, ... s_lock(3PPS)
sbrk: change private data segment — size ..... brk(3PPS)
getdtablesize: get descriptor table — size ..... getdtablesize(2SEQ)
getpagesize: get system page — size ..... getpagesize(2SEQ)
setdtablesize: set descriptor table — size ..... setdtablesize(2SEQ)
shsbrk: change shared data segment — size ..... shbrk(3PPS)
of object files — size: print section sizes ..... see *C and Lang Tools Binder*
interval — sleep: suspend execution for an ..... sleep(1)
interval — sleep: suspend execution for ..... sleep(3C)
s_init_lock, — s_lock, s_clock, s_unlock: initialize, lock/ ..... s_lock(3PPS)
ttyslot: find the — slot in the utmp file of the current user ..... ttyslot(3C)
consmem, — sm, ss: console memory drivers ..... consmem(7)
ssignal, gsignal: — software signals ..... ssignal(3C)
qsort: quicker — sort ..... qsort(3C)
tsort: topological — sort ..... tsort(1)
sort: — sort and/or merge files ..... sort(1)
— sort: sort and/or merge files ..... sort(1)
select or reject lines common to two — sorted files ..... comm(1)
bsearch: binary search a — sorted table ..... bsearch(3C)
an object file list: produce C — source listing from ..... see *C and Lang Tools Binder*
— sp: STREAMS pipe device ..... sp(7)
open file into the processs address — space ..... mmap(2SEQ)
unmap part of the processs address — space ..... munmap(2SEQ)
into processs virtual address — space ..... pmap(7)
brk, sbrk: change data segment — space allocation ..... brk(2)
ct: — spawn getty to a remote terminal ..... ct(1C)
memory efficient way vfork: — spawn new process in a virtual ..... vfork(2)
fspec: format — specification in text files ..... fspec(4)
signal: — specify what to do upon receipt of a signal ..... signal(2)
getty: set terminal type, modes, — speed, and line discipline ..... getty(1M)
uugetty: set terminal type, modes, — speed, and line discipline ..... uugetty(1M)
gettydefs: — speed and terminal settings used by getty ..... gettydefs(4)
find spelling errors — spell, hashmake, spellin, hashcheck: ..... spell(1)
file — spellhist: unmatched spelling words ..... spellhist(4)

spell, hashmake, spellin, hashcheck: find ..... spell(1)
hashmake, spellin, hashcheck: find spelling errors ..... spell(1)
spellhist: unmatched spelling words file ..... spellhist(4)
csplit: context split ..... csplit(1)
split: split a file into pieces ..... split(1)
fsplit: split f77, ratfor, or efl files ..... fsplit(1)
split: split a file into pieces ..... split(1)
uucleanup: uucp spool directory clean-up ..... uucleanup(1M)
printf, fprintf, sprintf: print formatted output ..... printf(3S)
data in a machine-independent/ sputl, sgetl: access long integer ..... sputl(3X)
square root/ exp, log, log10, pow, sqrt: exponential, logarithm, power, ... exp(3M)
sqrt: exponential, logarithm, power, square root functions ..... exp(3M)
rand, srand: simple random-number generator ..... rand(3C)
/lrand48, nrand48, mrand48, jrand48, srand48, seed48, lcong48: generate/ ..... drand48(3C)
consmem, sm, ss: console memory drivers ..... consmem(7)
scanf, fscanf, sscanf: convert formatted input ..... scanf(3S)
ssignal, gsignal: software signals ..... ssignal(3C)
wp: SSM parallel line printer interface ..... wp(7)
wd: SSM SCSI disk controller ..... wd(7)
SSM: System Services Module ..... ssm(7)
terminal multiplexor driver st: Systech MTI-800/1600/1650 ..... st(7)
sigstack: set and/or get signal stack context ..... sigstack(2SEQ)
copy: perform a standalone data transfer ..... copy(8)
prtvtoc: print the VTOC of a standalone disk device ..... prtvtoc(8)
dual-channel disk/ zdformat: standalone disk formatter for ..... zdformat(8)
CCSformat: standalone embedded SCSI disk formatter ..... ccsformat(8)
dump: standalone memory dump ..... dump(8)
operating system boot: run a standalone program or bring up the ..... boot(8)
intro: introduction to system standalone utilities ..... intro(8)
stdio: standard buffered input/output package ..... stdio(3S)
package stdipc: ftok: standard interprocess communication ..... stdipc(3C)
sh, rsh: shell, the standard/restricted command programming/ ..... sh(1)
operating system for beginning/ starter: information about the ..... starter(1)
lpsched, lpshut, lpmove: start/stop the LP print service and move/ ..... lpsched(1M)
/prctmp, prdaily, prtacct, shutacct, startup, turnacct: shell procedures/ ..... acctsh(1M)
call stat: data returned by stat system ..... stat(5)
stat, lstat, fstat: get file status ..... stat(2)
stat: data returned by stat system call ..... stat(5)
information statfs, fstatfs: get filesystem ..... statfs(2)
ustat: get filesystem statistics ..... ustat(2)
ff: list filenames and statistics for a filesystem ..... ff(1M)
fsstat: report filesystem status ..... fsstat(1M)
communication facilities status ..... ipcs(1)
ps: report process status ..... ps(1)
stat, lstat, fstat: get file status ..... stat(2)
feof, clearerr, fileno: stream status inquiries ..... ferror(3S)
uustat: uucp status inquiry and job control ..... uustat(1C)
lpstat: print information about the status of the LP print service ..... lpstat(1)
input/output package stdio: standard buffered ..... stdio(3S)
communication package stdipc: ftok: standard interprocess ..... stdipc(3C)
stime: set time ..... stime(2)
waitpid: wait for a child process to stop or terminate ..... wait(2)

lpsched, lpshut, lpmove: start/ stop the LP print service and move/.....lpsched(1M)
rc0: run commands performed to stop the operating system.....rc0(1M)
icheck: filesystem storage consistency check.....icheck(1M)
strace: print STREAMS trace messages.....strace(1M)
strncmp, strcpy, strncpy,/ string: strcat, strdup, strncat, strcmp,.....string(3C)
/strncmp, strcpy, strncpy, strlen, strchr, strrchr, strpbrk, strspn,/.....string(3C)
cleanup program strclean: STREAMS error logger.....strclean(1M)
string: strcat, strdup, strncat, strcmp, strncmp, strcpy, strncpy,/.....string(3C)
/strdup, strncat, strcmp, strncmp, strcpy, strncpy, strlen, strchr,/.....string(3C)
/strchr, strrchr, strpbrk, strspn, strcspn, strtok: string operations.....string(3C)
strcpy, strncpy,/ string: strcat, strdup, strncat, strcmp, strncmp,.....string(3C)
fclose, fflush: close or flush a stream.....fclose(3S)
fopen, freopen, fdopen: open a stream.....fopen(3S)
reposition a file pointer in a stream.....fseek(3S)
getw: get character or word from a stream.....getc(3S)
getmsg: get next message off a stream.....getmsg(2)
gets, fgets: get a string from a stream.....gets(3S)
putw: put character or word on a stream.....putc(3S)
putmsg: send a message on a stream.....putmsg(2)
puts, fputs: put a string on a stream.....puts(3S)
setvbuf: assign buffering to a stream.....setbuf(3S)
push character back into input stream.....ungetc(3S)
sed: stream editor.....sed(1)
ferror, feof, clearerr, fileno: stream status inquiries.....ferror(3S)
streamio: STREAMS ioctl commands.....streamio(7)
driver streampty: STREAMS pseudo terminal.....streampty(7)
clone: open any minor device on a STREAMS driver.....clone(7)
strclean: STREAMS error logger cleanup program.....strclean(1M)
strerr: STREAMS error logger daemon.....strerr(1M)
log: interface to STREAMS error logging and event tracing.....log(7)
poll: STREAMS input/output multiplexing.....poll(2)
streamio: STREAMS ioctl commands.....streamio(7)
Transport Interface cooperating STREAMS module.....timod(7)
Interface read/write interface STREAMS module.....tirdwr(7)
sp: STREAMS pipe device.....sp(7)
streampty: STREAMS pseudo terminal driver.....streampty(7)
strace: print STREAMS trace messages.....strace(1M)
streamtty: Generic STREAMS tty driver interface.....streamtty(7)
ld0: STREAMS tty line discipline.....ld0(7)
driver interface streamtty: Generic STREAMS tty.....streamtty(7)
strerr: STREAMS error logger daemon.....strerr(1M)
strftime: format time information.....strftime(3C)
long integer and base-64 ASCII string.....a64l(3C)
tzset: convert date and time to string.....ctime(3C)
convert floating-point number to string.....ecvt(3C)
fgrep: search a file for a character string.....fgrep(1)
regex: match patterns against a string.....regex(1F)
gps: graphical primitive string, format of graphical files.....gps(4)
gets, fgets: get a string from a stream.....gets(3S)
puts, fputs: put a string on a stream.....puts(3S)
echo: put string on virtual output.....echo(1F)
strpbrk, strspn, strcspn, strtok: string operations.....string(3C)

strcmp, strncmp, strcpy, strncpy,/ string: strcat, strdup, strncat, ...string(3C)
strtod, atof: convert string to double-precision number.....strtod(3C)
strtol, atol, atoi: convert string to integer.....strtol(3C)
cftime: language specific strings.....cftime(4)
object file strip: strip an.....see *C and Lang Tools Binder*
/strcmp, strncmp, strcpy, strncpy, strlen, strchr, strrchr, strpbrk,/.....string(3C)
string: strcat, strdup, strncat, strcmp, strncmp, strcpy, strncpy,/.....string(3C)
/strcat, strdup, strncat, strcmp, strncmp, strcpy, strncpy, strlen,/.....string(3C)
/strncat, strcmp, strncmp, strcpy, strncpy, strlen, strchr, strrchr,/.....string(3C)
/strncpy, strlen, strchr, strrchr, strpbrk, strspn, strcspn, strtok:/.....string(3C)
/strcpy, strncpy, strlen, strchr, strrchr, strpbrk, strspn, strcspn,/.....string(3C)
/strlen, strchr, strrchr, strpbrk, strspn, strcspn, strtok: string/.....string(3C)
character strstr: find first occurrence of.....strstr(3C)
double-precision number strtod, atof: convert string to.....strtod(3C)
strrchr, strpbrk, strspn, strcspn, strtok: string operations.....string(3C)
to integer strtol, atol, atoi: convert string.....strtol(3C)
processes using a file or file structure.....fuser(1M)
getscsimatch: get scsiinfo structure.....getscsiinfo(3SEQ)
getzdinfobyname: get zdinfo structure.....getzdinfo(3SEQ)
read constab entries into a structure.....read_constab(3SEQ)
t_alloc: allocate a library structure.....t_alloc(3N)
t_free: free a library structure.....t_free(3N)
state (defunct) stty, gtty: set and get terminal.....stty(2)
stty: set the options for a terminal.....stty(1)
su: become superuser or another user.....su(1M)
backup.codes1: backup subcode definitions.....backup_codes1(4)
m_fork: execute a subprogram in parallel.....m_fork(3PPS)
dbm_clearerr, dmb_blkflush: database subroutines.....ndbm(3X)
plot: graphics interface subroutines.....plot(3X)
sulog: log of substitute user events.....sulog(4)
sulog: log of substitute user events.....sulog(4)
of a file sum: print checksum and block count.....sum(1)
du: summarize disk usage.....du(1M)
repquota: summarize quotas for a filesystem.....repquota(1M)
records acctcms: command summary from per-process accounting .....acctcms(1M)
s_init_lock, s_lock, s_clock, s_unlock: initialize, lock, unlock locks.....s_lock(3PPS)
sync: update the superblock.....sync(1M)
sync: update superblock.....sync(2)
su: become superuser or another user.....su(1M)
getgroups: get supplementary group IDs.....getgroups(2)
uucp: networking support files.....uucp(4)
lptypes: table of printer types supported by the menu system.....lptypes(4)
m_park_procs, m_rele_procs: suspend and resume child process/.....m_park_procs(3PPS)
sleep: suspend execution for an interval.....sleep(1)
sleep: suspend execution for interval.....sleep(3C)
pause: suspend process until signal.....pause(2)
swab: swap bytes.....swab(3C)
wait at barrier s_init_barrier, s_wait_barrier: initialize barrier,.....s_wait_barrier(3PPS)
swap: swap administrative interface.....swap(1M)
swab: swap bytes.....swab(3C)
swap: swap administrative interface.....swap(1M)
/a command with priority aging, PFF, swapping, or processor migration/.....noage(1M)

seeks on DCC disks sweepon, sweepoff: enable/disable patrol ..... sweepon(1M)
patrol seeks on DCC disks sweepon, sweepoff: enable/disable ..... sweepon(1M)
symbol table/ ldgetname: retrieve symbol name for common object file ..... ldgetname(3X)
symbol name for common object file ldgetname: retrieve symbol table entry ..... ldgetname(3X)
ldtbindex: compute the index of a symbol table entry of a common/ ..... ldtbindex(3X)
ldtbread: read an indexed symbol table entry of a common object/ ..... ldtbread(3X)
syms: common object file symbol table format ..... syms(4)
ldtbseek: seek to the symbol table of a common object file ..... ldtbseek(3X)
unistd: file header for symbolic constants ..... unistd(4)
readlink: read value of a symbolic link ..... readlink(2SEQ)
symlink: make symbolic link to a file ..... symlink(2SEQ)
of common operating system terms and symbols ..... glossary(1)
file symlink: make symbolic link to a ..... symlink(2SEQ)
table format syms: common object file symbol ..... syms(4)
sync: update superblock ..... sync(2)
sync: update the superblock ..... sync(1M)
controlling FMLI object vsig: synchronize a co-process with the ..... vsig(1F)
t_sync: synchronize transport library ..... t_sync(3N)
check documents for correct eqn syntax ..... checkeq(1)
(command interpreter) with C-like syntax ..... csh(1)
administration sysadm: menu interface to do system ..... sysadm(1)
syscall: indirect system call ..... syscall(2)
variables sysconf: get configurable system ..... sysconf(2)
configuration file sys_config: format of a system ..... sys_config(4)
perror, errno, sys_errlist, sys_nerr: system error messages ..... perror(3C)
sysi86: machine specific functions ..... sysi86(2)
menu system functions syskeymap: identify keys to internal ..... syskeymap(4)
perror, errno, sys_errlist, sys_nerr: system error messages ..... perror(3C)
multiplexor driver st: Systech MTI-800/1600/1650 terminal ..... st(7)
lp: Systech parallel line printer interface ..... lp(7)
bsearch: binary search a sorted table ..... bsearch(3C)
constab: constructed devices table ..... constab(4)
devtab: virtual device table ..... devtab(4)
dltsectbl: refresh security table ..... dltsectbl(1M)
kernel auto-configuration device table ..... dumpconf(1M)
fstab: file-system- table ..... fstab(4)
mnttab: mounted filesystem table ..... mnttab(4)
setmnt: establish mount table ..... setmnt(1M)
shells: shell table ..... shells(4)
task: global security table ..... task(4)
name for common object file symbol table entry ..... ldgetname(3X)
/compute the index of a symbol table entry of a common object file ..... ldtbindex(3X)
ldtbread: read an indexed symbol table entry of a common object file ..... ldtbread(3X)
syms: common object file symbol table format ..... syms(4)
tasktbl: create security table in shared memory ..... tasktbl(1M)
get maximum in-use descriptor table index ..... getdtablemax(2SEQ)
ldtbseek: seek to the symbol table of a common object file ..... ldtbseek(3X)
the menu system lptypes: table of printer types supported by ..... lptypes(4)
getdtablesize: get descriptor table size ..... getdtablesize(2SEQ)
setdtablesize: set descriptor table size ..... setdtablesize(2SEQ)
classification and conversion tables ..... chrtbl(1M)
hdestroy: manage hash search tables ..... hsearch(3C)

tabs: set — tabs on a terminal ..... tabs(1)
— tabs: set tabs on a terminal ..... tabs(1)
— t_accept: accept a connect request ..... t_accept(3N)
ctags: create a — tags file ..... ctags(1)
file — tail: deliver the last part of a ..... tail(1)
structure — t_alloc: allocate a library ..... t_alloc(3N)
trig: sin, cos, — tan, asin, acos, atan, atan2: trigonometric/ ..... trig(3M)
sinh, cosh, — tanh: hyperbolic functions ..... sinh(3M)
tm: SCSI — tape controller ..... tm(7)
ts: SCSI — tape controller ..... ts(7)
hpio: Hewlett-Packard 2645A terminal — tape file archiver ..... hpio(1)
tar: — tape file archiver ..... tar(1)
mtio: UNIX magnetic — tape interface ..... mtio(7)
tg: SCSI 1/2-inch magnetic — tape interface ..... tg(7)
xt: Xylogics 472 MULITBUS magnetic — tape interface ..... xt(7)
mt: magnetic — tape manipulating program ..... mt(1)
— tar: tape file archiver ..... tar(1)
— task: global security table ..... task(4)
shared memory — tasktbl: create security table in ..... tasktbl(1M)
transport endpoint — t_bind: bind an address to a ..... t_bind(3N)
deroff: remove nroff, — tbl, and eqn constructs ..... deroff(1)
terminal state — tcgetattr, tcsetattr: get/set ..... tcgetattr(2)
group ID — tcgetpgrp: get foreground process ..... tcgetpgrp(2)
— t_close: close a transport endpoint ..... t_close(3N)
with another transport user — t_connect: establish a connection ..... t_connect(3N)
tcgetattr, — tcsetattr: get/set terminal state ..... tcgetattr(2)
group ID — tcsetpgrp: set foreground process ..... tcsetpgrp(2)
tsearch, tfind, — tdelete, twalk: manage binary search trees ..... tsearch(3C)
— tee: pipe fitting ..... tee(1)
4014: paginator for the — Tektronix 4014 terminal ..... 4014(1)
init, — telinit: process control initialization ..... init(1M)
directory: opendir, readdir, — telldir, seekdir, rewinddir,/ ..... directory(3X)
temporary file tmpnam, — tempnam: create a name for a ..... tmpnam(3S)
tmpfile: create a — temporary file ..... tmpfile(3S)
tmpnam, tempnam: create a name for a — temporary file ..... tmpnam(3S)
terminals — term: conventional names for ..... term(5)
term: format of compiled — term file ..... term(4)
— term: format of compiled term file ..... term(4)
description captoinfo: convert a — termcap description into a terminfo ..... captoinfo(1M)
base — termcap: terminal capability data ..... termcap(4)
paginator for the Tektronix 4014 — terminal ..... 4014(1)
special functions of the DASI 450 — terminal ..... 450(1)
ct: spawn getty to a remote — terminal ..... ct(1C)
ctermid: generate filename for — terminal ..... ctermid(3S)
hangup the specified controlling — terminal ..... fvhangup(2SEQ)
stty: set the options for a — terminal ..... stty(1)
tabs: set tabs on a — terminal ..... tabs(1)
tty: get the name of the — terminal ..... tty(1)
ttyname, isatty: find name of a — terminal ..... ttyname(3C)
/associated with a terminal or the — terminal associated with a device ..... ttydevname(1)
termcap: — terminal capability data base ..... termcap(4)
terminfo: — terminal capability database ..... terminfo(4)

streampty: STREAMS pseudo terminal driver.....streampty(7)
greek: select terminal filter.....greek(1)
tty: controlling terminal (gentty) interface.....tty(7)
termio: general terminal interface.....termio(7)
dial: establish an out-going terminal line connection.....dial(3C)
st: Systech MTI-800/1600/1650 terminal multiplexor driver.....st(7)
xs: Xylogics 781 terminal multiplexor driver.....xs(7)
tput: initialize a terminal or query terminfo database.....tput(1)
/get the device associated with a terminal or the terminal associated/.....ttydevname(1)
optimization package curses: terminal screen handling and.....curses(3X)
gettydefs: speed and terminal settings used by getty.....gettydefs(4)
tcgetattr, tcsetattr: get/set terminal state.....tcgetattr(2)
stty, gtty: set and get terminal state (defunct).....stty(2)
hpio: Hewlett-Packard 2645A terminal tape file archiver.....hpio(1)
line discipline getty: set terminal type, modes, speed, and.....getty(1M)
line discipline uugetty: set terminal type, modes, speed, and.....uugetty(1M)
functions of DASI 300 and 300s terminals.....300(1)
getpseudotty: get pseudo terminals.....getpseudotty(3SEQ)
special functions of Hewlett-Packard terminals.....hp(1)
term: conventional names for terminals.....term(5)
wait for a child process to stop or terminate.....wait(2)
kill: terminate a process.....kill(1)
exit, _exit: terminate process.....exit(2)
tic: terminfo compiler.....tic(1M)
tput: initialize a terminal or query terminfo database.....tput(1)
convert a termcap description into a terminfo description.....captoinfo(1M)
infocmp: compare or print out terminfo descriptions.....infocmp(1M)
database terminfo: terminal capability.....terminfo(4)
termio: general terminal interface.....termio(7)
of common operating system terms and symbols.....glossary(1)
t_error: produce error message.....t_error(3N)
test: condition evaluation command.....test(1)
isnan: isnand, isnanf: test for floating-point NaN/.....isnan(3C)
ed, red: text editor.....ed(1)
ex: text editor.....ex(1)
edit: text editor (variant of ex for casual users).....edit(1)
errormsg: error message text file.....errormsg(4)
newform: change the format of a text file.....newform(1)
fspec: format specification in text files.....fspec(4)
isismgr: ISIS file manager, updates text files in ISIS file format.....isismgr(1M)
plock: lock process, text, or data in memory.....plock(2)
search trees tsearch, tfind, tdelete, twalk: manage binary.....tsearch(3C)
reporter tfipc: interprocess communication.....tfipc(1M)
login shell tfmenush: assign menu system as user.....tfmenush(1M)
tfnotify: send completion message.....tfnotify(1M)
t_free: free a library structure.....t_free(3N)
interface tg: SCSI 1/2-inch magnetic tape.....tg(7)
service information t_getinfo: get protocol-specific.....t_getinfo(3N)
t_getstate: get the current state.....t_getstate(3N)
merge: three-way file merge.....merge(1)
tic: terminfo compiler.....tic(1M)
batch: execute commands at a later time.....at(1)

view a file one full screen at a time ..... more(1)
setting up an environment at login time ..... profile(4)
stime: set time ..... stime(2)
time: get time ..... time(2)
time, ptime: time a command ..... time(1)
and system activity timex: time a command; report process data ..... timex(1)
acctcon1, acctcon2: connect-time accounting ..... acctcon(1M)
time: get time ..... time(2)
strftime: format time information ..... strftime(3C)
profil: execution time profile ..... profil(2)
time, ptime: time a command ..... time(1)
mktime, tzset: convert date and time to string ..... ctime(3C)
clock: report CPU time used ..... clock(3C)
timezone: set default system time zone ..... timezone(4)
times: get process and child process times ..... times(2)
set file access and modification times ..... utime(2)
times: get process and child process ..... times(2)
update access and modification times of a file ..... touch(1)
process data and system activity timex: time a command; report ..... timex(1)
zone timezone: set default system time ..... timezone(4)
cooperating STREAMS module timod: Transport Interface ..... timod(7)
read/write interface STREAMS module tirdwr: Transport Interface ..... tirdwr(7)
request t_listen: listen for a connect ..... t_listen(3N)
a transport endpoint t_look: look at the current event on ..... t_look(3N)
tm: SCSI tape controller ..... tm(7)
processor tmp_affinity: bind a process to a ..... tmp_affinity(2SEQ)
tmp_ctl: multi-processor control ..... tmp_ctl(2SEQ)
tmpfile: create a temporary file ..... tmpfile(3S)
temporary file tmpnam, tempnam: create a name for a ..... tmpnam(3S)
/isgraph, isascii, tolower, toupper, toascci, _tolower, _toupper/ ..... ctype(3C)
/tolower, _toupper, _tolower, toascii: translate characters ..... conv(3C)
conv: toupper, tolower, _toupper, _tolower, toascii: translate/ ..... conv(3C)
/isascii, tolower, toupper, toascci, _tolower, _toupper, setchrclass:/ ..... ctype(3C)
/ispunct, isprint, isgraph, isascii, tolower, toupper, toascci, _tolower/ ..... ctype(3C)
conv: toupper, tolower, _toupper, _tolower, toascii: translate/ ..... conv(3C)
endpoint t_open: establish a transport ..... t_open(3N)
tsort: topological sort ..... tsort(1)
transport endpoint t_optmgmt: manage options for a ..... t_optmgmt(3N)
acctmerg: merge or add total accounting files ..... acctmerg(1M)
modification times of a file touch: update access and ..... touch(1)
/tolower, toupper, toascci, _tolower, _toupper, setchrclass: character/ ..... ctype(3C)
/isprint, isgraph, isascii, tolower, toupper, toascci, _tolower/ ..... ctype(3C)
conv: toupper, tolower, _toupper, _tolower, toascii: translate/ ..... conv(3C)
_tolower, toascii: translate/ conv: toupper, tolower, _toupper, ... conv(3C)
terminfo database tput: initialize a terminal or query ..... tput(1)
tr: translate characters ..... tr(1)
ptrace, mptrace: process trace facility ..... ptrace(2)
strace: print STREAMS trace messages ..... strace(1M)
to STREAMS error logging and event tracing ..... log(7)
copy: perform a standalone data transfer ..... copy(8)
init.d: system level transition files directory ..... init.d(4)
_toupper, _tolower, toascii: translate characters ..... conv(3C)

tr: translate characters ..... tr(1)
t_bind: bind an address to a transport endpoint ..... t_bind(3N)
t_close: close a transport endpoint ..... t_close(3N)
look at the current event on a transport endpoint ..... t_look(3N)
t_open: establish a transport endpoint ..... t_open(3N)
t_optmgmt: manage options for a transport endpoint ..... t_optmgmt(3N)
t_unbind: disable a transport endpoint ..... t_unbind(3N)
STREAMS module timod: Transport Interface cooperating ..... timod(7)
interface STREAMS module tirdwr: Transport Interface read/write ..... tirdwr(7)
t_sync: synchronize transport library ..... t_sync(3N)
the scheduler for the uucp file transport program ..... uusched(1M)
uucico: file transport program for the uucp system ..... uucico(1M)
establish a connection with another transport user ..... t_connect(3N)
data sent over a connection t_rcv: receive data or expedited ..... t_rcv(3N)
confirmation from a connect request t_rcvconnect: receive the ..... t_rcvconnect(3N)
disconnect t_rcvdis: retrieve information from ..... t_rcvdis(3N)
orderly release indication t_rcvrel: acknowledge receipt of an ..... t_rcvrel(3N)
t_rcvudata: receive a data unit ..... t_rcvudata(3N)
error indication t_rcvuderr: receive a unit data ..... t_rcvuderr(3N)
ftw: walk a file tree ..... ftw(3C)
tdelete, twalk: manage binary search trees ..... tsearch(3C)
atan, atan2: trigonometric/ trig: sin, cos, tan, asin, acos, ... trig(3M)
cos, tan, asin, acos, atan, atan2: trigonometric functions ..... trig(3M)
true, false: provide truth values ..... true(1)
truncate, ftruncate: truncate a file to a specified length ..... truncate(2SEQ)
to a specified length truncate, ftruncate: truncate a file ..... truncate(2SEQ)
u3b2, u3b5, vax: get processor type truth value ..... machid(1)
true, false: provide truth values ..... true(1)
ts: SCSI tape controller ..... ts(7)
manage binary search trees tsearch, tfind, tdelete, twalk: ..... tsearch(3C)
over a connection t_snd: send data or expedited data ..... t_snd(3N)
disconnect request t_snddis: send user-initiated ..... t_snddis(3N)
release t_sndrel: initiate an orderly ..... t_sndrel(3N)
t_sndudata: send a data unit ..... t_sndudata(3N)
tsort: topological sort ..... tsort(1)
library t_sync: synchronize transport ..... t_sync(3N)
interface tty: controlling terminal (gentty) ..... tty(7)
streamtty: Generic STREAMS tty driver interface ..... streamtty(7)
tty: get the name of the terminal ..... tty(1)
ld0: STREAMS tty line discipline ..... ld0(7)
between major/minor numbers and tty names ..... dev_to_ttyname(3C)
associated with a terminal or the/ ttydevname: get the device ..... ttydevname(1)
terminal ttyname, isatty: find name of a ..... ttyname(3C)
major/minor/ dev_to_ttyname, ttyname_to_dev: convert between ..... dev_to_ttyname(3C)
file of the current user ttyslot: find the slot in the utmp ..... ttyslot(3C)
endpoint t_unbind: disable a transport ..... t_unbind(3N)
parameters vmtune: tune system-wide virtual-memory ..... vmtune(1M)
tunefs: tune up an existing filesystem ..... tunefs(1M)
filesystem tunefs: tune up an existing ..... tunefs(1M)
examine and change virtual memory tuning parameters ..... vm_ctl(2SEQ)
/prdaily, prtacct, shutacct, startup, turnacct: shell procedures for/ ..... acctsh(1M)
tsearch, tfind, tdelete, twalk: manage binary search trees ..... tsearch(3C)

types: primitive system data — types.....types(5)
types: primitive system data types.....types(5)
lptypes: table of printer — types supported by the menu system.....lptypes(4)
/asctime, cftime, ascftime, mktime, — tzset: convert date and time to/.....ctime(3C)
machid: i386, ns32000, pdp11, — u3b, u3b2, u3b5, vax: get processor type/.....machid(1)
machid: i386, ns32000, pdp11, u3b, — u3b2, u3b5, vax: get processor type/.....machid(1)
/i386, ns32000, pdp11, u3b, u3b2, — u3b5, vax: get processor type truth/.....machid(1)
uadmin: administrative control .....uadmin(1M)
uadmin: administrative control.....uadmin(2)
/endspent, fgetspent, lckpwdf, — ulckpwdf: get shadow password file entry.....getspent(3X)
ulimit: get and set user limits .....ulimit(2)
mask — umask: set and get file creation.....umask(2)
umask: set file-creation mode mask.....umask(1)
mount, — umount: mount and unmount filesystems.....mount(1M)
umount: unmount a filesystem .....umount(2)
filesystems mountall, — umountall: mount, unmount multiple.....mountall(1M)
system — uname: get name of current operating.....uname(2)
uname: print name of current system .....uname(1)
expand data compress, — uncompress, zcat: compress and .....compress(1)
unget: — undo a previous get of an SCCS file .....unget(1)
SCCS file — unget: undo a previous get of an .....unget(1)
input stream — ungetc: push character back into .....ungetc(3S)
file — uniq: report repeated lines in a.....uniq(1)
mktemp: make a — unique filename.....mktemp(3C)
constants — unistd: file header for symbolic.....unistd(4)
t_rcvudata: receive a data — unit.....t_rcvudata(3N)
t_sndudata: send a data — unit.....t_sndudata(3N)
t_rcvuderr: receive a — unit data error indication.....t_rcvuderr(3N)
units: conversion program .....units(1)
mtio: — UNIX magnetic tape interface .....mtio(7)
cu: call another — UNIX system .....cu(1C)
link, unlink: link and — unlink files and directories.....link(1M)
link, — unlink: link and unlink files and directories.....link(1M)
unlink: remove directory entry.....unlink(2)
m_lock, m_unlock: lock, — unlock locks .....m_lock(3PPS)
s_clock, s_unlock: initialize, lock, — unlock locks.....s_lock(3PPS)
space munmap: — unmap part of the processs address .....munmap(2SEQ)
spellhist: — unmatched spelling words file.....spellhist(4)
umount: — unmount a filesystem.....umount(2)
mount, umount: mount and — unmount filesystems.....mount(1M)
mountall, umountall: mount, — unmount multiple filesystems.....mountall(1M)
pack, pcat, — unpack: compress and expand files.....pack(1)
set, unset: set and — unset environment variables in core or in files.....set(1F)
variables in core or in files set, — unset: set and unset environment .....set(1F)
lsearch, lfind: linear search and — update.....lsearch(3C)
touch: — update access and modification times of a file.....touch(1)
make: maintain, — update, and regenerate groups of programs .....make(1)
bdflush: — update daemon.....bdflush(1M)
bdflush: — update daemon.....bdflush(2SEQ)
sync: — update superblock.....sync(2)
sync: — update the superblock.....sync(1M)
from/ pwconv: installs and — updates /etc/shadow with information.....pwconv(1M)

isismgr: ISIS file manager, updates text files in ISIS file format ..... isismgr(1M)
been up uptime: show how long system has ..... uptime(1)
du: summarize disk usage ..... du(1M)
quota: display disk usage and limits ..... quota(1)
retrieve a command description and usage examples ..... usage(1)
description and usage examples usage: retrieve a command ..... usage(1)
usclk: 32-bit microsecond counter ..... usclk(7)
getusclk, usclk_init: get/init microsecond clock ..... getusclk(3SEQ)
get character login name of the user ..... cuserid(3S)
logname: return login name of user ..... logname(3X)
setup: initialize system for first user ..... setup(1)
su: become superuser or another user ..... su(1M)
a connection with another transport user ..... t_connect(3N)
slot in the utmp file of the current user ..... ttyslot(3C)
write: write to another user ..... write(1)
cpset: copy file and set mode, user, and group IDs ..... cpset(1)
setuid, setgid: set user and group IDs ..... setuid(2)
id: print user and group IDs and names ..... id(1M)
crontab: user crontab file ..... crontab(1)
addusr.def: add user defaults ..... addusr_def(4)
/geteuid, getgid, getegid: get real user, effective user, real group,/ ..... getuid(2)
environ: user environment ..... environ(5)
run commands performed for multi- user environment ..... rc2(1M)
sulog: log of substitute user events ..... sulog(4)
generate disk accounting data by user ID ..... diskusg(1M)
ulimit: get and set user limits ..... ulimit(2)
tfmenush: assign menu system as user login shell ..... tfmenush(1M)
menumsg: route user message to menu message queue ..... menumsg(1M)
edquota: edit user quotas ..... edquota(1M)
/getegid: get real user, effective user, real group, and effective/ ..... getuid(2)
t_snddis: send user-initiated disconnect request ..... t_snddis(3N)
editor (variant of ex for casual users) ..... edit(1)
the operating system for beginning users ..... starter(1)
wall: write to all users ..... wall(1)
ut_delete_user: add/delete users in utmp file ..... utadduser(3X)
mail, rmail: send mail to users or read mail ..... mail(1)
login attempts /usr/adm/loginlog: log of failed/ ..... loginlog(4)
ustat: get filesystem statistics ..... ustat(2)
add/delete users in utmp file ut_add_user, ut_delete_user: ..... utadduser(3X)
utmp file ut_add_user, ut_delete_user: add/delete users in ..... utadduser(3X)
introduction to system standalone utilities ..... intro(8)
get_process_stats: get resource utilization information ..... get_process_stats(2SEQ)
modification times utime: set file access and ..... utime(2)
utmp, wtmp: utmp and wtmp entry formats ..... utmp(4)
ut_delete_user: add/delete users in utmp file ..... utadduser(3X)
endutent, utmpname, getutpid: access utmp file entry ..... getut(3C)
ttyslot: find the slot in the utmp file of the current user ..... ttyslot(3C)
formats utmp, wtmp: utmp and wtmp entry ..... utmp(4)
/pututline, setutent, endutent, utmpname, getutpid: access utmp file/ ..... getut(3C)
and permissions file uucheck: check the uucp directories ..... uucheck(1M)
the uucp system uucico: file transport program for ..... uucico(1M)
clean-up uucleanup: uucp spool directory ..... uucleanup(1M)

uucheck: check the uucp directories and permissions file.....uucheck(1M)
uusched: the scheduler for the uucp file transport program.....uusched(1M)
uucp: networking support files.....uucp(4)
uucleanup: uucp spool directory clean-up.....uucleanup(1M)
uustat: uucp status inquiry and job control.....uustat(1C)
file transport program for the uucp system.....uucico(1M)
UNIX-to-UNIX system/ uucp, uulog, uuname:.....uucp(1C)
speed, and line discipline uugetty: set terminal type, modes,.....uugetty(1M)
uucp, uulog, uuname: UNIX-to-UNIX/.....uucp(1C)
uucp, uulog, uuname: UNIX-to-UNIX system copy.....uucp(1C)
uuto, uupick: public UNIX-to-UNIX/.....uuto(1C)
file transport program uusched: the scheduler for the uucp.....uusched(1M)
control uustat: uucp status inquiry and job.....uustat(1C)
UNIX-to-UNIX system/ uuto, uupick: public.....uuto(1C)
with debugging on Uutry: try to contact remote system.....uutry(1M)
system command execution uux: UNIX-to-UNIX.....uux(1C)
requests uuxqt: execute remote command.....uuxqt(1M)
val: validate SCCS file.....val(1)
val: validate SCCS file.....val(1)
valloc: aligned memory allocator.....valloc(3SEQ)
abs: return integer absolute value.....abs(3C)
u3b5, vax: get processor type truth value,.....machid(1)
getenv: return value for environment name.....getenv(3C)
floor, ceiling, remainder, absolute value functions.....floor(3M)
readlink: read value of a symbolic link.....readlink(2SEQ)
putenv: change or add value to environment.....putenv(3C)
current form field to its default values.....reset(1F)
true, false: provide truth values.....true(1)
values: machine-dependent values.....values(5)
values: machine-dependent values.....values(5)
print formatted output of a varargs argument list.....vprintf(3S)
list varargs: handle variable argument.....varargs(5)
varargs: handle variable argument list.....varargs(5)
fpathconf: get configurable pathname variables.....pathconf(2)
sysconf: get configurable system variables.....sysconf(2)
.environ, .pref, .variables: FACE files.....env(4)
unset: set and unset environment variables in core or in files.....set(1F)
edit: text editor (variant of ex for casual users).....edit(1)
proc_ctl: control over various process attributes.....proc_ctl(2SEQ)
/ns32000, pdp11, u3b, u3b2, u3b5, vax: get processor type truth value.....machid(1)
vc: version control.....vc(1)
get option letter from argument vector.....getopt(3C)
readv: read vector from file.....readv(2SEQ)
writev: write vector on a file.....writev(2SEQ)
assert: verify program assertion.....assert(3X)
vc: version control.....vc(1)
get: get a version of an SCCS file.....get(1)
sccsdiff: compare two versions of an SCCS file.....sccsdiff(1)
virtual memory efficient way vfork: spawn new process in a.....vfork(2)
output of a varargs/ vprintf, vfprintf, vsprintf: print formatted.....vprintf(3S)
editor based on ex vi: screen-oriented (visual) display.....vi(1)
more: view a file one full screen at a time.....more(1)

map physical addresses into processs — virtual address space ..... pmap(7)
devtab: — virtual device table ..... devtab(4)
devdestroy: build and destroy — virtual devices ..... devbuild(1M)
vfork: spawn new process in a — virtual memory efficient way ..... vfork(2)
vm_ctl: examine and change — virtual memory tuning parameters ..... vm_ctl(2SEQ)
echo: put string on — virtual output ..... echo(1F)
controlling terminal fvhangup: — virtually hangup the specified ..... fvhangup(2SEQ)
vmtune: tune system-wide — virtual-memory parameters ..... vmtune(1M)
pmap: display or alter parameters of — virtual-to-physical mapping devices ..... pmap(1M)
vi: screen-oriented — visual display editor based on ex ..... vi(1)
memory tuning parameters — vm_ctl: examine and change virtual ..... vm_ctl(2SEQ)
virtual-memory parameters — vmtune: tune system-wide ..... vmtune(1M)
filesystem — volcopy: make literal copy of ..... volcopy(1M)
fs, inode: format of file system — volume ..... fs(4)
formatted output of a varargs/ — vprintf, vfprintf, vsprintf: print ..... vprintf(3S)
the controlling FMLI object — vsig: synchronize a co-process with ..... vsig(1F)
a varargs/ vprintf, vfprintf, — vsprintf: print formatted output of ..... vprintf(3S)
prtvtoc: print the — VTOC of a block device ..... prtvtoc(1M)
prtvtoc: print the — VTOC of a standalone disk device ..... prtvtoc(8)
mkvtoc: populate — VTOC on disks ..... mkvtoc(1M)
vtoc: — VTOC partitioning driver ..... vtoc(7)
vtoc: VTOC partitioning driver ..... vtoc(7)
s_wait_barrier: initialize barrier, — wait at barrier ..... s_wait_barrier(3PPS)
wait: await completion of process ..... wait(1)
wait, waitpid: — wait for a child process to stop or terminate ..... wait(2)
sigsuspend: — wait for a signal ..... sigsuspend(2)
process to stop or terminate — wait, waitpid: wait for a child ..... wait(2)
stop or terminate wait, — waitpid: wait for a child process to ..... wait(2)
ftw: — walk a file tree ..... ftw(3C)
wall: write to all users ..... wall(1)
wc: word count ..... wc(1)
wd: SSM SCSI disk controller ..... wd(7)
what: identify SCCS files ..... what(1)
whatis: describe what a command is ..... whatis(1)
whodo: — who is doing what ..... whodo(1M)
who: — who is on the system ..... who(1)
who: who is on the system ..... who(1)
whodo: who is doing what ..... whodo(1M)
vmtune: tune system- — wide virtual-memory parameters ..... vmtune(1M)
addformopt: add a form/SELECT — window option to a menu ..... addformopt(1M)
wc: — word count ..... wc(1)
fgetc, getw: get character or — word from a stream ..... getc(3S)
fputc, putw: put character or — word on a stream ..... putc(3S)
spellhist: unmatched spelling — words file ..... spellhist(4)
cd: change — working directory ..... cd(1)
chdir: change — working directory ..... chdir(2)
getcwd: get pathname of current — working directory ..... getcwd(3C)
pwd: — working directory name ..... pwd(1)
specific alarms and/or the — working indicator ..... indicator(1F)
interface — wp: SSM parallel line printer ..... wp(7)
lseek: move read/ — write file pointer ..... lseek(2)
tirdwr: Transport Interface read/ — write interface STREAMS module ..... tirdwr(7)

write: write on a file ..... write(2)
putpwent: write password file entry ..... putpwent(3C)
putspent: write shadow password file entry ..... putspent(3X)
wall: write to all users ..... wall(1)
write: write to another user ..... write(1)
writev: write vector on a file ..... writev(2SEQ)
write: write on a file ..... write(2)
write: write to another user ..... write(1)
writev: write vector on a file ..... writev(2SEQ)
open: open for reading or writing ..... open(2)
utmp, wtmp: utmp and wtmp entry formats ..... utmp(4)
utmp, wtmp: utmp and wtmp entry formats ..... utmp(4)
accounting records fwtmp, wtmpfix: manipulate connect ..... fwtmp(1M)
and execute command xargs: construct argument list(s) ..... xargs(1)
multiplexor driver xs: Xylogics 781 terminal ..... xs(7)
tape interface xt: Xylogics 472 MULITBUS magnetic ..... xt(7)
xt: Xylogics 472 MULITBUS magnetic tape interface ..... xt(7)
xs: Xylogics 781 terminal multiplexor driver ..... xs(7)
bessel: j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
bessel: j0, j1, jn, y0, y1, yn: Bessel functions ..... bessel(3M)
yacc: yet another compiler-compiler ..... yacc(1)
compress, uncompress, zcat: compress and expand data ..... compress(1)
zd: dual-channel disk controller ..... zd(7)
for dual-channel disk controller zdformat: standalone disk formatter ..... zdformat(8)
getzdinfo, getzdinfobyname: get zdinfo structure ..... getzdinfo(3SEQ)

NAME

intro – introduction to system calls and error numbers

SYNOPSIS

#include <errno.h>

DESCRIPTION

This section describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value. This is almost always –1 or the **NULL** pointer; the individual descriptions specify the details. An error number is also made available in the external variable **errno**. **errno** is not cleared on successful calls, so it should be tested only after an error has been indicated.

Each system call description attempts to list all possible error numbers in alphabetical order. The following is a complete list of the error numbers in numerical order, and their names as defined in *<errno.h>:*

1 EPERM "Not owner" Typically this error indicates an attempt to modify a file in some way forbidden except to its owner or superuser. It is also returned for attempts by ordinary users to do things allowed only to the superuser.

2 ENOENT "No such file or directory"
This error occurs when a filename is specified and the file should exist but doesn't, or when one of the directories in a pathname does not exist.

3 ESRCH "No such process"
No process can be found corresponding to that specified by **pid** in **kill**(2) or **ptrace**(2).

4 EINTR "Interrupted system call"
An asynchronous signal (such as interrupt or quit), which the user has elected to catch, occurred during a system call. If execution is resumed after processing the signal, it will appear as if the interrupted system call returned this error condition.

5 EIO "I/O error" Some physical I/O error has occurred. This error may in some cases occur on a call following the one to which it actually applies.

6 ENXIO "No such device or address"
I/O on a special file refers to a subdevice which does not exist, or beyond the limits of the device. It may also occur when, for example, a tape drive is not on-line or no disk pack is loaded on a drive.

7 E2BIG "Arg list too long"
An argument list longer than ARG_MAX bytes is presented to a member of the **exec**(2) family.

8 ENOEXEC "Exec format error"
A request is made to execute a file which, although it has the appropriate permissions, does not start with a valid magic number [see **a.out**(4)].

9 EBADF "Bad file number"
Either a file descriptor isn't open, or a **read**(2) [respectively, **write**(2)] request is made to a file which is open only for writing (respectively, reading).

10 ECHILD "No child processes"
A **wait** was executed by a process that had no existing or unwaited-for child processes.

11 EAGAIN "No more processes"
A **fork** failed because the system's process table is full or the user is not allowed to create any more processes, or a system call failed because of insufficient swap space.

12 ENOMEM "Not enough space"
During an **exec**(2), **brk**(2), or **sbrk**(2), a program asks for more space than the system is able to supply. This is not a temporary condition; the maximum space size is a system parameter.

13 EACCES "Permission denied"
An attempt was made to access a file in a way forbidden by the protection system.

14 EFAULT "Bad address"
The system encountered a hardware fault in attempting to use an argument of a system call.

15 ENOTBLK "Block device required"
A non-block file was mentioned where a block device was required, e.g., in **mount**(2).

16 EBUSY "Device or resource busy"
An attempt was made to mount a device that was already mounted or an attempt was made to dismount a device on which there is an active file (open file, current directory, mounted-on file, active text segment). It will also occur if an attempt is made to enable accounting when it is already enabled. The device or resource is currently unavailable.

17 EEXIST "File exists" An existing file was mentioned in an inappropriate context, e.g., **link**(2).

19 ENODEV "No such device"
An attempt was made to apply an inappropriate system call to a device; e.g., read a write-only device.

20 ENOTDIR "Not a directory"
A non-directory was specified where a directory is required, for example in a path prefix or as an argument to **chdir**(2).

21 EISDIR "Is a directory"
An attempt was made to write on a directory.

22 EINVAL "Invalid argument"
Some invalid argument (e.g., dismounting a

non-mounted device; mentioning an undefined signal in **signal**(2) or **kill**(2); reading or writing a file for which **lseek**(2) has generated a negative pointer). Also set by the math functions described in the (3M) entries of this manual.

23 ENFILE "File table overflow"
The system file table is full, and temporarily no more **opens** can be accepted.

24 EMFILE "Too many open files"
No process may have more than NOFILES (default 20) descriptors open at a time. Refer to NOFILES later in this introduction.

25 ENOTTY "Not a character device (or) Not a typewriter"
An attempt was made to **ioctl**(2) a file that is not a character-special device.

26 ETXTBSY "Text file busy"
An attempt was made to execute a shared-text program that is currently open for writing. Also an attempt to open for writing or to remove a shared-text program that is being executed.

27 EFBIG "File too large"
The size of a file exceeded the maximum file size or ULIMIT (see **ulimit**(2)).

28 ENOSPC "No space left on device"
During a **write**(2) to an ordinary file, or the creation of a symbolic link, or a directory, there is no free space left on the file system. During the creation of a directory, a special file, a symbolic link or a link to a file, the directory in which the entry for the new file is being placed cannot be extended because there is no space left on the file system containing that directory. During the creation of a file system object, there are no more inodes available in that file system. In **fcntl**(2), the setting or removing of record locks on a file cannot be accomplished because there are no more record entries left on the system.

29 ESPIPE "Illegal seek"
An **lseek**(2) was issued to a pipe.

30 EROFS "Read-only filesystem"
An attempt to modify a file or directory was made on a device mounted read-only.

31 EMLINK "Too many links"
An attempt to make more than the maximum number of links (default MAXLINK, see *<sys/param.h>)* to a file.

32 EPIPE "Broken pipe" A write on a pipe for which there is no process to read the data. This condition normally generates a signal; the error is returned if the signal is ignored.

33 EDOM "Math argument"
The argument of a function in the math package (3M) is out of the domain of the function.

34 ERANGE "Result too large"
The value of a function in the math package (3M) is not representable within machine precision.

35 ENOMSG "No message of desired type"
An attempt was made to receive a message of a type that does not exist on the specified message queue [see **msgop**(2)].

36 EIDRM "Identifier removed"
This error is returned to processes that resume execution due to the removal of an identifier from the filesystem's name space [see **msgctl**(2), **semctl**(2), and **shmctl**(2)].

37-44 Reserved numbers

45 EDEADLK "Deadlock"
A deadlock situation was detected and avoided. This error pertains to file and record locking.

46 ENOLCK "No lock"
In **fcntl**(2) the setting or removing of record locks on a file cannot be accomplished because there are no more record entries left on the system.

60 ENOSTR "Not a stream"
A **putmsg**(2) or **getmsg**(2) system call was attempted on a file descriptor that is not a STREAMS device.

62 ETIME "Stream ioctl timeout"
The timer set for a STREAMS **ioctl**(2) call has expired. The cause of this error is device specific and could indicate either a hardware or software failure, or perhaps a timeout value that is too short for the specific operation. The status of the **ioctl**(2) operation is indeterminate.

63 ENOSR "No stream resources"
The requested operation cannot be completed currently due to the exhaustion of internal STREAMS resources.

65 ENOPKG "No package"
This error occurs when users attempt to use a system call from a package which has not been installed.

77 EBADMSG "Bad message"
During a **read**(2), **getmsg**(2), or **ioctl**(2) I_RECVFD system call to a STREAMS device, an invalid message has come to the head of the queue. Examples are:

read(2) - control information or a passed file descriptor.

getmsg(2) - passed file descriptor.

ioctl(2) - control or data information.

88 ENAMETOOLONG "File name too long"
A component of a path name exceeded NAME_MAX characters, or an entire path name exceeded PATH_MAX characters. A path name containing symbolic links may also produce this error if the symbolic link expansion exceeds PATH_MAX characters or contains a component bigger than NAME_MAX characters.

89 OOP "Too many levels of symbolic links"
A path name lookup involved more than *n* symbolic links, where *n* is binary configurable (default is 8).

The following symbolic constants are used to describe various system limits in the following sections. The actual values can be found in *<limits.h>*.

ARG_MAX
Maximum length of arguments for the **exec**(2) function in bytes, including environment data

CHILD_MAX
Maximum number of simultaneous processes per real user ID

NAME_MAX
Maximum number of bytes in a filename (the count excludes a terminating null)

PATH_MAX
Maximum number of bytes in a pathname (the count excludes a terminating null)

PIPE_BUF
Maximum number of bytes which can be written atomically to a pipe

NGROUPS_MAX
Maximum number of simultaneous supplementary group IDs per process.

The following symbolic constants are used to describe various system limits in the following sections. The actual values can be found in *<sys/param.h>*.

MAXPID
The maximum process ID

MAXLINK
The maximum number of links to a file

MAXUID
The maximum user ID

DEFINITIONS

Process ID. Each active process in the system is uniquely identified by a positive integer called a process ID. The range of this ID is from 1 to MAXPID.

Parent Process ID. A new process is created by a currently active process [see **fork**(2)]. The parent process ID of a process is the process ID of its creator.

Process Group ID. Each active process is a member of a process group that is

identified by a positive integer called the process group ID. This ID is the process ID of the group leader. This grouping permits the signaling of related processes [see **kill**(2)].

Session Group. Each active process is a member of a session group. This grouping permits the exchange of a foreground process group and its related job-control mechanisms.

Foreground Process Group ID. Each active process can be a member of a process group that is identified by a positive integer called the forground process group. ID. This grouping is used to terminate a group of related processes upon termination of one of the processes in the group [see **exit**(2) and **signal**(2)].

Real User ID and Real Group Each user allowed on the system is identified by a positive integer (0 to MAXUID) called a real user ID.

Each user is also a member of a group. The group is identified by a positive integer called the real group ID.

An active process has a real user ID and real group ID that are set to the real user ID and real group ID, respectively, of the user responsible for the creation of the process.

Effective User ID and Effective Group An active process has an effective user ID and an effective group ID that are used to determine file access permissions (see below). The effective user ID and effective group ID are equal to the process's real user ID and real group ID respectively, unless the process or one of its ancestors evolved from a file that had the set-user-ID bit or set-group ID bit set [see **exec**(2)].

Supplementary Group ID. Each user on the system can be a member of NGROUPS_MAX supplementary groups (in addition to the group ID). Supplementary groups are used along with the effective group ID in determining file access privileges. They are established when a user logs in [see **login**(1)] to the system. The file */etc/group* lists all the supplementary groups and their members.

Superuser. A process is recognized as a superuser process and is granted special privileges, such as immunity from file permissions, if its effective user ID is 0.

Special Processes. The processes with a process ID of 0 and a process ID of 1 are special processes and are referred to as **proc0** and **proc1**.

proc0 is the scheduler. **proc1** is the initialization process (**init**). **proc1** is the ancestor of every other process in the system and is used to control the process structure.

File Descriptor. A file descriptor is a small integer used to do I/O on a file. The value of a file descriptor is from 0 to (NOFILES - 1). A process may have no more than NOFILES. file descriptors open simultaneously. A file descriptor is returned by system calls such as **open**(2), or **pipe**(2). The file descriptor is used as an argument by calls such as **read**(2), **write**(2), **ioctl**(2), and **close**(2).

NOFILES. This is the maximum number of file descriptors that a process may have open simultaneously. NOFILES is set when the kernel is configured; the default operating system value for NOFILES is 20. This number can be changed by reconfiguring the kernel (refer to the *System Administration Guide*). Every process, on creation, can have up to 20 open file descriptors. In order for a process to get more than 20 file descriptors, the process must call **setdtablesize**(2seq). The argument to **setdtablesize** cannot exceed the value of NOFILES. The current value of the file descriptor table size for a process can be determined using the **getdtablesize**(2seq) system call. The **getdtablemax**(2seq) system call returns the highest numbered file descriptor currently opened.

File Name. Names consisting of 1 to NAME_MAX characters may be used to name an ordinary file, special file or directory.

These characters may be selected from the set of all character values excluding \0 (null) and the ASCII code for / (slash).

Note that it is generally unwise to use *, ?, [, or] as part of filenames because of the special meaning attached to these characters by the shell [see **sh**(1)]. Although permitted, the use of unprintable characters in filenames should be avoided.

Path Name and Path Prefix. A pathname is a null-terminated character string starting with an optional slash (/), followed by zero or more directory names separated by slashes, optionally followed by a filename. The total length of a path name must be less than PATH_MAX characters.

If a pathname begins with a slash, the path search begins at the root directory. Otherwise, the search begins from the current working directory.

A slash by itself names the root directory.

Unless specifically stated otherwise, the null pathname is treated as if it named a non-existent file.

Directory. A directory is a special type of file containing entries which are references to other files. Directory entries are called links. By convention, a directory contains at least two links, . and .., referred to as *dot* and *dot-dot* respectively. Dot refers to the directory itself and dot-dot refers to its parent directory.

Root Directory and Current Working Directory. Each process has associated with it a concept of a root directory and a current working directory for the purpose of resolving pathname searches. The root directory of a process

need not be the root directory of the root filesystem.

File Access Permissions. Every file in the file system has a set of access permissions. These permissions are used in determining whether a process may perform a requested operation on the file (such as opening a file for writing). Access permissions are established at the time a file is created. They may be changed at some later time through the **chmod**(2) call.

File access is broken down according to whether a file may be: read, written, or executed. Directory files use the execute permission to control if the directory may be searched.

File access permissions are interpreted by the system as they apply to three different classes of user: the owner of the file, those users in the file's group, anyone else. Every file has an independent set of access permissions for each of these classes. When an access check is made, the system decides if permission should be granted by checking the access information applicable to the caller.

Read, write, and execute/search permissions on a file are granted to a process if one or more of the following are true:

The effective user ID of the process is superuser.

The effective user ID of the process matches the user ID of the owner of the file and the appropriate access bit of the owner portion (0700) of the file mode is set.

The effective user ID of the process does not match the user ID of the owner of the file, and the effective group ID of the process, or one of its supplementary group IDs, matches the group of the file and the appropriate access bit of the group portion (0070) of the file mode is set.

The effective user ID of the process does not match the user ID of the owner of the file, and the effective group ID of the process, or one of its supplementary group IDs, does not match the group ID of the file, and the appropriate access bit of the other portion (0007) of the file mode is set.

Otherwise, the corresponding permissions are denied.

Message Queue Identifier. A message queue identifier (**msqid**) is a unique positive integer created by a **msgget**(2) system call. Each msqid has a message queue and a data structure associated with it. The data structure is referred to as **msqid_ds** and contains the following members:

```
struct ipc_perm msg_perm;
struct msg *msg_first;
struct                    msg *msg_last;
ushort msg_cbytes;
ushort msg_qnum;
ushort msg_qbytes;
pid_t  msg_lspid;
pid_t  msg_lrpid;
time_t msg_stime;
```

```
time_t msg_rtime;
time_t msg_ctime;
```

msg_perm is an **ipc_perm** structure that specifies the message operation permission (see below). This structure includes the following members:

```
uid_t  cuid;      /* creator user id */
gid_t  cgid;      /* creator group id */
uid_t  uid;       /* user id */
gid_t  gid;       /* group id */
ushort mode;      /* r/w permission */
ushort seq;       /* slot usage sequence # */
key_t  key;       /* key */
```

msg *msg_first is a pointer to the first message on the queue.

msg *msg_last is a pointer to the last message on the queue.

msg_cbytes is the current number of bytes on the queue.

msg_qnum is the number of messages currently on the queue.

msg_qbytes is the maximum number of bytes allowed on the queue.

msg_lspid is the process ID of the last process that performed a **msgsnd** operation.

msg_lrpid is the process ID of the last process that performed a **msgrcv** operation.

msg_stime is the time of the last **msgsnd** operation.

msg_rtime is the time of the last **msgrcv** operation

msg_ctime is the time of the last **msgctl**(2) operation that changed a member of the above structure.

Message Operation Permissions. In the **msgop**(2) and **msgctl**(2) system call descriptions, the permission required for an operation is given as "{token}", where "token" is the type of permission needed, interpreted as follows:

00400	Read by owner
00200	Write by owner
00040	Read by group
00020	Write by group
00004	Read by others
00002	Write by others

Read and write permissions on a **msqid** are granted to a process if one or more of the following are true:

The effective user ID of the process is superuser.

The effective user ID of the process matches **msg_perm.cuid** or **msg_perm.uid** in the data structure associated with **msqid** and the appropriate bit of the user portion (0600) of **msg_perm.mode** is set.

The effective group ID of the process matches **msg_perm.cgid** or **msg_perm.gid** and the appropriate bit of the group portion (060) of

msg_perm.mode is set.

The appropriate bit of the other portion (006) of **msg_perm.mode** is set.

Otherwise, the corresponding permissions are denied.

Semaphore Identifier. A semaphore identifier (**semid**) is a unique positive integer created by a **semget**(2) system call. Each **semid** has a set of semaphores and a data structure associated with it. The data structure is referred to as **semid_ds** and contains the following members:

```
struct ipc_perm sem_perm;/* operation permission struct */
struct sem *sem_base;    /* ptr to first semaphore in set */
ushort sem_nsems;        /* number of sems in set */
time_t sem_otime;        /* last operation time */
time_t sem_ctime;        /* last change time */
                         /* Times measured in secs since */
                         /* 00:00:00 UTC, Jan. 1, 1970 */
```

sem_perm is an **ipc_perm** structure that specifies the semaphore operation permission (see below). This structure includes the following members:

```
uid_t  uid;      /* user id */
gid_t  gid;      /* group id */
uid_t  cuid;     /* creator user id */
gid_t  cgid;     /* creator group id */
ushort mode;     /* r/a permission */
ushort seq;      /* slot usage sequence number */
key_t  key;      /* key */
```

sem_nsems is equal to the number of semaphores in the set. Each semaphore in the set is referenced by a positive integer referred to as a **sem_num**. **sem_num** values run sequentially from 0 to the value of sem_nsems minus 1.

sem_otime is the time of the last **semop**(2) operation.

sem_ctime is the time of the last **semctl**(2) operation that changed a member of the above structure.

A semaphore is a data structure called **sem** that contains the following members:

```
ushort semval;   /* semaphore value */
pid_t  sempid;   /* pid of last operation  */
ushort semncnt;  /* # awaiting semval > cval */
ushort semzcnt;  /* # awaiting semval = 0 */
```

semval is a non-negative integer which is the actual value of the semphore.

sempid is equal to the process ID of the last process that performed a semaphore operation on this semaphore.

semncnt is a count of the number of processes that are currently suspended awaiting this semaphore's semval to become greater than its current value.

semzcnt is a count of the number of processes that are currently suspended awaiting this semaphore's semval to become zero.

Semaphore Operation Permissions.

In the **semop**(2) and **semctl**(2) system call descriptions, the permission required for an operation is given as "{token}", where "token" is the type of permission needed interpreted as follows:

00400	Read by owner
00200	Alter by owner
00040	Read by group
00020	Alter by group
00004	Read by others
00002	Alter by others

Read and alter permissions on a semid are granted to a process if one or more of the following are true:

The effective user ID of the process is superuser.

The effective user ID of the process matches **sem_perm.cuid** or **sem_perm.uid** in the data structure associated with *semid* and the appropriate bit of the user portion (0600) of **sem_perm.mode** is set.

The effective group ID of the process matches **sem_perm.cgid** or **sem_perm.gid** and the appropriate bit of the group portion (060) of **sem_perm.mode** is set.

The appropriate bit of the other portion (006) of **sem_perm.mode** is set.

Otherwise, the corresponding permissions are denied.

Shared Memory Identifier. A shared memory identifier **shmid**, is a unique positive integer created by a **shmget**(2) system call. Each **shmid** has a segment of memory (referred to as a shared memory segment) and a data structure associated with it. (Note that these shared memory segments must be explicitly removed by the user after the last reference to them is removed.) The data structure is referred to as **shmid_ds** and contains the following members:

```
struct ipc_perm shm_perm;/* operation permission struct */
int    shm_segsz;        /* size of segment */
struct region *shm_reg;  /*ptr to region structure */
char   pad[4];           /* for swap compatibility */
pid_t  shm_lpid;         /* pid of last operation */
pid_t  shm_cpid;         /* creator pid */
ushort shm_nattch;       /* number of current attaches */
ushort shm_cnattch;      /* used only for shminfo */
time_t shm_atime;        /* last attach time */
```

```
time_t shm_dtime;       /* last detach time */
time_t shm_ctime;       /* last change time */
                        /* Times measured in secs since */
                        /* 00:00:00 UTC, Jan. 1, 1970 */
```

shm_perm is an **ipc_perm** structure that specifies the shared memory operation permission (see below). This structure includes the following members:

```
uid_t  cuid;    /* creator user id */
gid_t  cgid;    /* creator group id */
uid_t  uid;     /* user id */
gid_t  gid;     /* group id */
ushort mode;    /* r/w permission */
ushort seq;     /* slot usage sequence # */
key_t  key;     /* key */
```

shm_segsz specifies the size of the shared memory segment in bytes.

shm_cpid is the process id of the process that created the shared memory identifier.

shm_lpid is the process id of the last process that performed a **shmop**(2) operation. This field is currently not maintained on a **shmdt**(2) operation.

shm_nattch is the number of processes that currently have this segment attached.

shm_atime is the time of the last **shmat**(2) operation,

shm_dtime is the time of the last **shmdt**(2) operation. This field is currently not maintained.

shm_ctime is the time of the last **shmctl**(2) operation that changed one of the members of the above structure.

Shared Memory Operation Permissions. In the **shmop**(2) and **shmctl**(2) system call descriptions, the permission required for an operation is given as "{token}", where "token" is the type of permission needed interpreted as follows:

00400	Read by owner
00200	Write by owner
00040	Read by group
00020	Write by group
00004	Read by others
00002	Write by others

Read and write permissions on a **shmid** are granted to a process if one or more of the following are true:

The effective user ID of the process is superuser.

The effective user ID of the process matches **shm_perm.cuid** or **shm_perm.uid** in the data structure associated with **shmid** and the appropriate bit of the user portion (0600) of **shm_perm.mode** is set.

The effective group ID of the process matches **shm_perm.cgid** or **shm_perm.gid** and the appropriate bit of the group portion (060) of **shm_perm.mode** is set.

The appropriate bit of the other portion (06) of **shm_perm.mode** is set.

Otherwise, the corresponding permissions are denied.

STREAMS. A set of kernel mechanisms that support the development of network services and data communication drivers. It defines interface standards for character input/output within the kernel and between the kernel and user level processes. The STREAMS mechanism is composed of utility routines, kernel facilities and a set of data structures.

Stream. A stream is a full-duplex data path within the kernel between a user process and driver routines. The primary components are a stream head, a driver and zero or more modules between the stream head and driver. A stream is analogous to a shell pipeline except that data flow and processing are bidirectional.

Stream Head. In a stream, the stream head is the end of the stream that provides the interface between the stream and a user process. The principle functions of the stream head are processing STREAMS-related system calls, and passing data and information between a user process and the stream.

Driver. In a stream, the driver provides the interface between peripheral hardware and the stream. A driver can also be a pseudodriver, such as a multiplexor or log driver [see **log**(7)], which is not associated with a hardware device.

Module. A module is an entity containing processing routines for input and output data. It always exists in the middle of a stream, between the stream's head and a driver. A module is the STREAMS counterpart to the commands in a shell pipeline except that a module contains a pair of functions which allow independent bidirectional (downstream and upstream) data flow and processing.

Downstream. In a stream, the direction from stream head to driver.

Upstream. In a stream, the direction from driver to stream head.

Message. In a stream, one or more blocks of data or information, with associated STREAMS control structures. Messages can be of several defined types, which identify the message contents. Messages are the only means of transferring data and communicating within a stream.

Message Queue. In a stream, a linked list of messages awaiting processing

by a module or driver.

Read Queue. In a stream, the message queue in a module or driver containing messages moving upstream.

Write Queue. In a stream, the message queue in a module or driver containing messages moving downstream.

Multiplexor. A multiplexor is a driver that allows streams associated with several user processes to be connected to a single driver, or several drivers to be connected to a single user process. STREAMS does not provide a general multiplexing driver, but does provide the facilities for constructing them, and for connecting multiplexed configurations of streams.

SEE ALSO

intro(3)

NAME

access – determine accessibility of a file

SYNOPSIS

int access (path, amode)
char *path;
int amode;

DESCRIPTION

path points to a pathname naming a file. **access** checks the named file for accessibility according to the bit pattern contained in **amode**, using the real user ID in place of the effective user ID and the real group ID in place of the effective group ID, and the supplementary group ID. The bit pattern contained in **amode** is constructed as follows:

04	read
02	write
01	execute (search)
00	check existence of file

Access to the file is denied if one or more of the following are true:

[EACCES]	Search permission is denied on a component of the path prefix.
[EACCES]	Permission bits of the file mode do not permit the requested access.
[EFAULT]	**path** points outside the allocated address space for the process.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENOENT]	Read, write, or execute (search) permission is requested for a null pathname.
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOTDIR]	A component of the path prefix is not a directory.
[ENOENT]	The named file does not exist.
[EROFS]	Write access is requested for a file on a read-only filesystem.
[ETXTBSY]	Write access is requested for a pure procedure (shared text) file that is being executed.

The owner of a file has permission checked with respect to the *owner* read, write, and execute mode bits. Members of the file's group other than the owner have permissions checked with respect to the *group* mode bits, and all others have permissions checked with respect to the *other* mode bits. A process is a member of a file's group if the group ID appears in its supplementary group list or is its effective group ID.

SEE ALSO

chmod(2), stat(2)

DIAGNOSTICS

If the requested access is permitted, a value of 0 is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error.

NAME

acct – enable or disable process accounting

SYNOPSIS

int acct (path)
char *path;

DESCRIPTION

acct is used to enable or disable the system process accounting routine. If the routine is enabled, an accounting record will be written on an accounting file for each process that terminates. Termination can be caused by one of two things: an **exit** call or a signal [see **exit**(2) and **signal**(2)]. The effective user ID of the calling process must be superuser to use this call.

path points to a pathname naming the accounting file. The accounting file format is given in **acct**(4).

The accounting routine is enabled if **path** is non-zero and no errors occur during the system call. It is disabled if **path** is zero and no errors occur during the system call.

acct will fail if one or more of the following are true:

[EACCES]	The file named by **path** is not an ordinary file.
[EBUSY]	An attempt is being made to enable accounting when it is already enabled.
[EFAULT]	**path** points to an illegal address.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	One or more components of the accounting file pathname do not exist.
[ENOTDIR]	A component of the path prefix is not a directory.
[EPERM]	The effective user of the calling process is not superuser.
[EROFS]	The named file resides on a read-only filesystem.

SEE ALSO

exit(2), signal(2), acct(4)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

alarm – set a process alarm clock

SYNOPSIS

unsigned alarm (sec)
unsigned sec;

DESCRIPTION

alarm instructs the alarm clock of the calling process to send the signal SIGALRM to the calling process after the number of real time seconds specified by **sec** have elapsed [see **signal**(2)].

Alarm requests are not stacked; successive calls reset the alarm clock of the calling process.

If **sec** is 0, any previously made alarm request is canceled.

If the process does a **fork**(2), the alarm request is cleared in the child process. Alarm requests remain pending across the **exec**(2) family of system calls.

SEE ALSO

fork(2), exec(2), pause(2), signal(2), sigpause(2), sigset(2), sigaction(2)

DIAGNOSTICS

alarm returns the amount of time previously remaining in the alarm clock of the calling process.

NAME

brk, sbrk – change data segment space allocation

SYNOPSIS

```
int brk (endds)
char *endds;

char *sbrk (incr)
int incr;
```

DESCRIPTION

brk and **sbrk** are used to change dynamically the amount of space allocated for the calling process's data segment (see **exec**(2)). The change is made by resetting the process's break value and allocating the appropriate amount of space. The break value is the address of the first location beyond the end of the data segment. The amount of allocated space increases as the break value increases. Newly allocated space is set to zero. If, however, the same memory space is reallocated to the same process its contents are undefined.

brk sets the break value to **endds** and changes the allocated space accordingly.

sbrk adds **incr** bytes to the break value and changes the allocated space accordingly. **incr** can be negative, in which case the amount of allocated space is decreased.

brk and **sbrk** fails without making any change in the allocated space if one or more of the following are true:

[EAGAIN] Swap space required to back up the memory being allocated is temporarily unavailable.

[EINVAL] If the change would force text to be deleted.

[ENOMEM] Such a change would result in more space being allocated than is allowed by the system-imposed maximum process size (see **ulimit**(2)), or the new data space would overlap with previously allocated memory (see **shmop**(2), **mmap**(2seq)).

SEE ALSO

exec(2), mmap(2seq), shmop(2), ulimit(2), end(3C)

DIAGNOSTICS

Upon successful completion, **brk** returns a value of 0 and **sbrk** returns the old break value. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

chdir – change working directory

SYNOPSIS

int chdir (path)
char *path;

DESCRIPTION

path points to the pathname of a directory. **chdir** causes the named directory to become the current working directory, the starting point for path searches for pathnames not beginning with / (slash).

chdir will fail and the current working directory will be unchanged if one or more of the following are true:

[EACCES]	Search permission is denied for any component of the path name.
[EFAULT]	**path** points outside the allocated address space of the process.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	The named directory does not exist.
[ENOTDIR]	A component of the pathname is not a directory.

SEE ALSO

chroot(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

chmod – change mode of file

SYNOPSIS

```
#include <sys/types.h>
int chmod (path, mode)
char *path;
mode_t mode;
```

DESCRIPTION

path points to a path name naming a file. **chmod** sets the access permission portion of the named file's mode according to the bit pattern contained in **mode**.

Access permission bits are interpreted as follows:

04000	Set user ID on execution.
020#0	Set group ID on execution if # is **7**, **5**, **3**, or **1** Enable mandatory file/record locking if # is **6**, **4**, **2**, or **0**
01000	Save text image after execution.
00400	Read by owner.
00200	Write by owner.
00100	Execute (search if a directory) by owner.
00070	Read, write, execute (search) by group.
00007	Read, write, execute (search) by others.

The effective user ID of the process must match the owner of the file or be superuser to change the mode of a file.

If the effective user ID of the process is not superuser and the file is not a directory, mode bit 01000 (save text image on execution) is cleared.

If the effective user ID of the process is not superuser and if the group ID of the file does not match the effective group ID of the process or one of the supplementary group IDs of the process, the mode bit 02000 (set group ID on execution) is cleared. Clearing the set group ID bit may disable mandatory record locking on that file.

If a 410 executable file has the sticky bit (mode bit 01000) set, the operating system will not delete the program text from the swap area when the last user process terminates. If a 413 executable file has the sticky bit set, the operating system will not delete the program text from memory when the last user process terminates. In either case, if the sticky bit is set the text will already be available (either in a swap area or in memory) when the next user of the file executes it, thus making execution faster.

If the executing process is not owned by the superuser, **chmod** masks the sticky-bit but does not return an error.

If a directory is writable and has the sticky bit set, files within that directory can be removed only if one or more of the following is true (see **unlink**(2)):

the user owns the file

the user owns the directory
the file is writable by the user
the user is the superuser

If the mode bit 02000 (set group ID on execution) is set and the mode bit 00010 (execute or search by group) is not set, mandatory file/record locking will exist on a regular file. This may affect future calls to **open**(2), **creat**(2), **read**(2), **write**(2), and **mmap(2seq)** on this file.

chmod will fail and the file mode will be unchanged if one or more of the following are true:

[EACCES]	Search permission is denied on a component of the path prefix.
[EFAULT]	**path** points outside the allocated address space of the process.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	The named file does not exist.
[ENOTDIR]	A component of the path prefix is not a directory.
[EPERM]	The effective user ID does not match the owner of the file and the effective user ID is not superuser.
[EROFS]	The named file resides on a read-only file system.

SEE ALSO

chmod(1), chown(2), creat(2), fcntl(2), mknod(2), open(2), read(2), write(2), mmap(2seq)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

chown – change owner and group of a file

SYNOPSIS

```
int chown (path, owner, group)
char *path;
uid_t owner;
gid_t group;
```

DESCRIPTION

path points to a pathname naming a file. The owner ID and group ID of the named file are set to the numeric values contained in **owner** and **group** respectively.

Only processes with effective user ID equal to the superuser may change the ownership of a file.

If **owner** or **group** is specified as (uid_t)-1 or (gid_t)-1, the corresponding ID of the file remains unchanged.

If **chown** is invoked by other than the superuser, the set-user-ID and set-group-ID bits of the file mode, 04000 and 02000 respectively, will be cleared.

chown will fail and the owner and group of the named file will remain unchanged if one or more of the following are true:

[EACCES]	Search permission is denied on a component of the path prefix.
[EFAULT]	**path** points outside the allocated address space of the process.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	The named file does not exist.
[ENOTDIR]	A component of the path prefix is not a directory.
[EPERM]	The effective user ID does not match the owner of the file and the effective user ID is not superuser.
[EROFS]	The named file resides on a read-only filesystem.

SEE ALSO

chown(1), chmod(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

chroot – change root directory

SYNOPSIS

int chroot (path)
char *path;

DESCRIPTION

path points to a pathname naming a directory. **chroot** causes the named directory to become the root directory, the starting point for path searches for pathnames beginning with /. The user's working directory is unaffected by the **chroot** system call.

The effective user ID of the process must be superuser to change the root directory.

The .. entry in the root directory is interpreted to mean the root directory itself. Thus, .. cannot be used to access files outside the subtree rooted at the root directory.

chroot will fail and the root directory will remain unchanged if one or more of the following are true:

[EFAULT]	**path** points outside the allocated address space of the process.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	The named directory does not exist.
[ENOTDIR]	Any component of the pathname is not a directory.
[EPERM]	The effective user ID is not superuser.

SEE ALSO

chdir(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

close – close a file descriptor

SYNOPSIS

int close (fildes)
int fildes;

DESCRIPTION

fildes is a file descriptor obtained from a **creat, open, dup, fcntl,** or **pipe** system call. **close** closes the file descriptor indicated by **fildes.** All outstanding record locks owned by the process (on the file indicated by **fildes**) are removed. Existing maps created by **mmap**(2seq) using this file descriptor remain mapped.

If a STREAMS file (see **intro**(2)) is closed, and the calling process had previously registered to receive a SIGPOLL signal (see **signal**(2), **sigset**(2), and **sigaction**(2)) for events associated with that file (see I_SETSIG in **streamio**(7)), the calling process is unregistered for events associated with the file. The last **close** for a stream causes the stream associated with **fildes** to be dismantled. If O_NDELAY is not set and there have been no signals posted for the stream, **close** waits up to 15 seconds, for each module and driver, for any output to drain before dismantling the stream. If the O_NDELAY flag is set or if there are any pending signals, **close** does not wait for output to drain, and dismantles the stream immediately.

The named file is closed unless one or more of the following are true:

[EBADF] **fildes** is not a valid open file descriptor.

[EINTR] A signal was caught during the **close** system call.

SEE ALSO

creat(2), dup(2), exec(2), fcntl(2), intro(2), open(2), pipe(2), signal(2), sigset(2), sigaction(2), streamio(7)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

creat – create a new file or rewrite an existing one

SYNOPSIS

```
#include <sys/types.h>
int creat (path, mode)
char *path;
mode_t mode;
```

DESCRIPTION

creat creates a new ordinary file or prepares to rewrite an existing file named by the pathname pointed to by **path**.

If the file exists, the length is truncated to 0 and the mode and owner are unchanged. Otherwise, the file's owner ID is set to the effective user ID of the process, the group ID of the file is set to the group ID of the directory in which this file is being created and the low-order 12 bits of the file mode are set to the value of **mode** modified as follows:

All bits set in the process's file mode creation mask are cleared (see **umask**(2)).

The `save text image after execution bit` of the mode is cleared (see **chmod**(2)).

If the effective user ID of the process is not superuser and if the group ID of the file does not match the effective group ID of the process or one of the supplementary group IDs of the process, the mode bit `set group ID on execution` is cleared (see **chmod**(2)). Clearing this bit may disable mandatory record locking on this file.

Upon successful completion, a write-only file descriptor is returned and the file is open for writing, even if the mode does not permit writing. The file pointer is set to the beginning of the file. The file descriptor is set to remain open across **exec** system calls (see **fcntl**(2)). No process may have more than NOFILES files open simultaneously (see **intro**(2)). A new file may be created with a mode that forbids writing.

creat fails if one or more of the following are true:

[EACCES]	Search permission is denied on a component of the path prefix.
[EACCES]	The file does not exist and the directory in which the file is to be created does not permit writing.
[EACCES]	The file exists and write permission is denied.
[EAGAIN]	The file exists, mandatory file/record locking is set, and there are outstanding record locks on the file (see **chmod**(2)).
[EFAULT]	**path** points outside the allocated address space of the process.
[EINTR]	A signal was caught during the **creat** system call.
[EIO]	An I/O error occurred while reading from or writing to the file system.

[EISDIR]	The named file is an existing directory.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[EMFILE]	NOFILES file descriptors are currently open. (see **intro**(2)).
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENFILE]	The system file table is full.
[ENOENT]	A component of the path prefix does not exist.
[ENOENT]	The pathname is null.
[ENOSPC]	There are no free inodes on the file system on which the file is being created.
[ENOSPC]	The directory in which the entry for the new file is being placed cannot be extended because there is no space left on the file system containing the directory.
[ENOTDIR]	A component of the path prefix is not a directory.
[EROFS]	The named file resides or would reside on a read-only filesystem.
[ETXTBSY]	The file is a pure procedure (shared text) file that is being executed.

SEE ALSO

chmod(2), close(2), dup(2), fcntl(2), intro(2), lseek(2), open(2), read(2), umask(2), write(2), getdtablesize(2seq)

DIAGNOSTICS

Upon successful completion, a non-negative integer, namely the file descriptor, is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

dup – duplicate an open file descriptor

SYNOPSIS

int dup (fildes)
int fildes;

DESCRIPTION

fildes is a file descriptor obtained from a **creat, open, dup, fcntl,** or **pipe** system call. **dup** returns a new file descriptor having the following in common with the original:

Same open file (or pipe).

Same file pointer (i.e., both file descriptors share one file pointer).

Same access mode (read, write or read/write).

The new file descriptor is set to remain open across **exec** system calls (see **fcntl**(2)).

The file descriptor returned is the lowest one available.

dup fails if one or more of the following are true:

[EBADF] **fildes** is not a valid open file descriptor.

[EMFILE] NOFILES file descriptors are currently open. (see **intro**(2)).

SEE ALSO

close(2), creat(2), exec(2), fcntl(2), intro(2), open(2), pipe(2), getdtablesize(2seq), lockf(3C)

DIAGNOSTICS

Upon successful completion a non-negative integer, namely the file descriptor, is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

dup2 – duplicate an open file descriptor

SYNOPSIS

int dup2 (fildes, fildes2)
int fildes, fildes2;

DESCRIPTION

fildes is a file descriptor referring to an open file, and **fildes2** is a non-negative integer less than **NOFILES** (see **intro**(2)). **dup2** causes **fildes2** to refer to the same file as **fildes**. If **fildes2** already referred to an open file, it is closed first.

dup2 fails if any of the following are true:

[EBADF] **fildes** is not a valid open file descriptor, or filedes2 is greater than NOFILES.

SEE ALSO

creat(2), close(2), exec(2), fcntl(2), intro(2), open(2), pipe(2), lockf(3C)

DIAGNOSTICS

Upon successful completion a non-negative integer, namely the file descriptor, is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

exec: execl, execv, execle, execve, execlp, execvp – execute a file

SYNOPSIS

```
int execl (path, arg0, arg1, ..., argn, (char *)0)
char *path, *arg0, *arg1, ..., *argn;

int execv (path, argv)
char *path, *argv[ ];

int execle (path, arg0, arg1, ..., argn, (char *)0, envp)
char *path, *arg0, *arg1, ..., *argn, *envp[ ];

int execve (path, argv, envp)
char *path, *argv[ ], *envp[ ];

int execlp (file, arg0, arg1, ..., argn, (char *)0)
char *file, *arg0, *arg1, ..., *argn;

int execvp (file, argv)
char *file, *argv[ ];
```

DESCRIPTION

exec in all its forms transforms the calling process into a new process. The new process is constructed from an ordinary, executable file called the *new process file*. This file consists of a header (see **a.out**(4)), a text segment, and a data segment. The data segment contains an initialized portion and an uninitialized portion (bss). There can be no return from a successful **exec** because the calling process is overlaid by the new process.

An interpreter file begins with a line of the form *#! interpreter [arg]*. Up to one argument can be specified on the initial line and it will apply to all invocations of the command. Additional command line arguments can be specified on each invocation. When an interpreter file is **exec**ed, the system loads the specified interpreter and gives it the name of the originally **exec**ed file as an argument, shifting over the rest of the arguments.

When a C program is executed, it is called as follows:

```
main (argc, argv, envp)
 int argc;
 char **argv, **envp;
```

where **argc** is the argument count, **argv** is an array of character pointers to the arguments themselves, and **envp** is an array of character pointers to the environment strings. As indicated, **argc** is conventionally at least one and the first member of the array points to a string containing the name of the file.

path points to a path name that identifies the new process file.

file points to the new process file. The path prefix for this file is obtained by a search of the directories passed as the environment line "PATH =" (see **environ**(5)). The environment is supplied by the shell (see **sh**(1)).

arg0, arg1, ..., argn are pointers to null-terminated character strings. These strings constitute the argument list available to the new process. By convention, at least **arg0** must be present and point to a string that is the same as **path** (or its last component).

Argv is an array of character pointers to null-terminated strings. These strings constitute the argument list available to the new process. By convention, **argv** must have at least one member, and it must point to a string that is the same as **path** (or its last component). **argv** is terminated by a null pointer.

envp is an array of character pointers to null-terminated strings. These strings constitute the environment for the new process. **envp** is terminated by a null pointer. For **execl** and **execv**, the C run-time start-off routine places a pointer to the environment of the calling process in the global cell:

```
extern char **environ;
```

and it is used to pass the environment of the calling process to the new process.

File descriptors open in the calling process remain open in the new process, except for those whose close-on-exec flag is set; see **fcntl**(2). For those file descriptors that remain open, the file pointer is unchanged.

Signals set to terminate the calling process will be set to terminate the new process. Signals set to be ignored by the calling process will be set to be ignored by the new process. Signals set to be caught by the calling process will be set to terminate the new process; see **signal**(2).

For signals set by **sigset**(2), **exec** will ensure that the new process has the same system signal action for each signal type whose action is SIG_DFL, SIG_IGN, or SIG_HOLD as the calling process. However, if the action is to catch the signal, then the action will be reset to SIG_DFL, and any pending signal for this type will be held.

If the set-user-ID mode bit of the new process file is set (see **chmod**(2)), **exec** sets the effective user ID of the new process to the owner ID of the new process file. Similarly, if the set-group-ID mode bit of the new process file is set, the effective group ID of the new process is set to the group ID of the new process file. If the new process file is an interpreter file, that is, it begins with a *#!*, the effective user and group IDs remain unchanged. This effectively disallows setuid shell scripts. The real user ID and real group ID of the new process, as well as the supplementary group ID, remain the same as those of the calling process.

Shared memory segments and mapped segments attached to the calling process will not be attached to the new process (see **shmop**(2) and **mmap**(2seq)).

Profiling is disabled for the new process; see **profil**(2).

The new process also inherits the following attributes from the calling process:

nice value (see **nice**(2))

process ID

parent process ID

process group ID

session ID

semadj values (see **semop**(2))

foreground process ID (see **exit**(2) and **signal**(2))

trace flag (see **ptrace**(2) request 0)

time left until an alarm clock signal (see **alarm**(2))

current working directory

root directory

file mode creation mask (see **umask**(2))

file size limit (see **ulimit**(2))

utime, stime, cutime, and **cstime** (see **times**(2))

file-locks (see **fcntl**(2) and **lockf**(3C))

affinity (see **tmp_affinity**(2))

exec fails and returns to the calling process if one or more of the following are true:

[EACCES]	Search permission is denied for a directory listed in the new process file's path prefix.
[EACCES]	The new process file is not an ordinary file.
[EACCES]	The new process file mode denies execution permission.
[EAGAIN]	The memory required to execute the new image exceeds the available space in the swap area.
[EFAULT]	**path, argv,** or **envp** point to an illegal address.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** is longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	One or more components of the new process path name of the file do not exist.
[ENOEXEC]	The **exec** is not an **execlp** or **execvp,** and the new process file has the appropriate access permission but an invalid magic number in its header.
[ENOMEM]	The new process requires more memory than is allowed by the system-imposed maximum.
[ENOTDIR]	A component of the new process path of the file prefix is not a directory.
[ETXTBSY]	The new process file is a pure procedure (shared text) file that is currently open for writing by some process.

[E2BIG] The number of bytes in the new process's argument list is greater than the system-imposed limit of ARG_MAX bytes.

SEE ALSO

sh(1), alarm(2), exit(2), fcntl(2), fork(2), nice(2), ptrace(2), semop(2), signal(2), sigset(2), times(2), ulimit(2), umask(2), lockf(3C), a.out(4), environ(5)

DIAGNOSTICS

If **exec** returns to the calling process an error has occurred; the return value is –1 and **errno** is set to indicate the error.

NAME

exit, _exit – terminate process

SYNOPSIS

void exit (status)
int status;
void _exit (status)
int status;

DESCRIPTION

exit terminates the calling process with the following consequences:

All of the file descriptors open in the calling process are closed.

If the parent process of the calling process is executing a **wait**, it is notified of the calling process's termination and the low order eight bits (i.e., bits 0377) of **status** are made available to it [see **wait**(2)].

If the parent process of the calling process is not executing a **wait**, the calling process is transformed into a zombie process. A *zombie process* is a process that only occupies a slot in the process table. It has no other space allocated either in user or kernel space. The process table slot that it occupies is partially overlaid with time accounting information (see *<sys/proc.h>*) to be used by **times** and **get_process_stats(2).**

The parent process ID of all of the calling processes' existing child processes and zombie processes is set to 1. This means the initialization process [see **intro**(2)] inherits each of these processes.

Each attached shared memory segment is detached and the value of **shm_nattach** in the data structure associated with its shared memory identifier is decremented by 1.

All mapped regions (see **mmap**(2seq)) are detached from the process.

For each semaphore for which the calling process has set a **semadj** value (see **semop**(2)), that **semadj** value is added to the **semval** of the specified semaphore.

If the process has a process, text, or data lock, an **unlock** is performed (see **plock**(2)).

An accounting record is written on the accounting file if the system's accounting routine is enabled (see **acct**(2)).

If the process is the controlling process (session leader with a controlling terminal), then SIGHUP signal is sent to each process in the current foreground process group.

A death-of-child signal is sent to the parent.

The C function **exit** may cause cleanup actions before the process exits. The function **_exit** circumvents all cleanup.

SEE ALSO

acct(2), intro(2), plock(2), semop(2), signal(2), sigset(2), sigaction(2seq), wait(2), get_process_stats(2)

WARNING

See **WARNING** in **signal**(2)

DIAGNOSTICS

None. There can be no return from an **exit** system call.

NAME

fcntl – file control

SYNOPSIS

#include <fcntl.h>

int fcntl (fildes, cmd, arg)
int fildes, cmd, arg;

DESCRIPTION

fcntl provides for control over open files. **fildes** is an open file descriptor obtained from a **creat, open, dup, fcntl,** or **pipe** system call.

The commands available are:

F_DUPFD — Return a new file descriptor as follows:

Lowest numbered available file descriptor greater than or equal to **arg**.

Same open file (or pipe) as the original file.

Same file pointer as the original file (i.e., both file descriptors share one file pointer).

Same access mode (read, write or read/write).

Same file status flags (i.e., both file descriptors share the same file status flags).

The close-on-exec flag associated with the new file descriptor is set to remain open across **exec**(2) system calls.

F_GETFD — Get the close-on-exec flag associated with the file descriptor **fildes**. If the low-order bit is 0, the file remains open across **exec**, otherwise the file is closed upon execution of **exec**.

F_GETFL — Get **file** status flags.

F_GETLK — Get the first lock which blocks the lock description given by the variable of type `struct flock` pointed to by **arg**. The information retrieved overwrites the information passed to **fcntl** in `struct flock`. If no lock is found that would prevent this lock from being created, then the structure is passed back unchanged except for the lock type which is set to F_UNLCK.

F_SETFD — Set the close-on-exec flag associated with **fildes** to the low-order bit of **arg** (0 or 1 as above).

F_SETFL — Set **file** status flags to **arg**. Only certain flags can be set (see **fcntl**(5)).

F_SETLK — Set or clear a file segment lock according to the variable of type `struct flock` pointed to by **arg** (see **fcntl**(5)). The **cmd** F_SETLK is used to establish read (F_RDLCK) and write (F_WRLCK) locks, as well as remove either type of lock (F_UNLCK). If a read or write lock cannot be set **fcntl** returns immediately with an error value of –1.

F_SETLKW — This **cmd** is the same as F_SETLK except that if a read or write lock is blocked by other locks, the process sleeps until

the segment is free to be locked.

A read lock prevents any process from write locking the protected area. More than one read lock may exist for a given segment of a file at a given time. The file descriptor on which a read lock is being placed must have been opened with read access.

A write lock prevents any process from read locking or write locking the protected area. Only one write lock may exist for a given segment of a file at a given time. The file descriptor on which a write lock is being placed must have been opened with write access.

The structure `flock` describes the type (**l_type**), starting offset (**l_whence**), relative offset (**l_start**), size (**l_len**), and process ID (**l_pid**) of the segment of the file to be affected. The process ID and system ID fields are used only with the F_GETLK **cmd** to return the values for a blocking lock. On DYNIX systems, **l_sysid** always has a value of zero. Locks may start and extend beyond the current end of a file, but may not be negative relative to the beginning of the file. A lock may be set to always extend to the end of file by setting **l_len** to zero (0). If such a lock also has **l_whence** and **l_start** set to zero (0), the whole file is locked. Changing or unlocking a segment from the middle of a larger locked segment leaves two smaller segments for either end. Locking a segment that is already locked by the calling process causes the old lock type to be removed and the new lock type to take effect. All locks associated with a file for a given process are removed when a file descriptor for that file is closed by that process or the process holding that file descriptor terminates. Locks are not inherited by a child process in a **fork**(2) system call.

When mandatory file and record locking is active on a file, (see **chmod**(2)), **read** and **write** system calls issued on the file are affected by the record locks in effect. The **mmap**(2seq) system call fails on any file which has mandatory locking enabled. This is to prevent a user from using mmap to modify a file while a mandatory lock asserted on that file.

fcntl fails if one or more of the following are true:

[EACCES]	**cmd** is F_SETLK the type of lock **l_type** is a read (F_RDLCK) lock and the segment of a file to be locked is already write locked by another process or the type is a write (F_WRLCK) lock and the segment of a file to be locked is already read or write locked by another process.
[EBADF]	**fildes** is not a valid open file descriptor.
[EDEADLK]	**cmd** is F_SETLKW, the lock is blocked by some lock from another process, and putting the calling-process to sleep, waiting for that lock to become free, would cause a deadlock.
[EFAULT]	**cmd** is F_SETLK, **arg** points outside the program address space.
[EINTR]	A signal was caught during the **fcntl system call.**
[EINVAL]	**cmd** is F_DUPFD. **arg** is either negative, or greater than or equal to ENOFILES (see **getdtablesize**(2) and **intro**(2)).

[EINVAL] **cmd** is F_GETLK, F_SETLK, or SETLKW and **arg** or the data it points to is not valid.

[EMFILE] **cmd** is F_DUPFD and NOFILES file descriptors are currently open.

[ENOLCK] **cmd** is F_SETLK or F_SETLKW, the type of lock is a read or write lock, and there are no more record locks available (too many file segments locked) because the system maximum has been exceeded.

SEE ALSO

close(2), creat(2), dup(2), exec(2), fork(2), intro(2), open(2), pipe(2), fcntl(5), mmap(2seq), getdtablesize(2seq)

DIAGNOSTICS

Upon successful completion, the value returned depends on **cmd** as follows:

F_DUPFD	A new file descriptor
F_GETFD	Value of flag (only the low-order bit is defined)
F_GETFL	Value of file flags
F_GETLK	Value other than –1
F_SETFD	Value other than –1
F_SETFL	Value other than –1
F_SETLK	Value other than –1
F_SETLKW	Value other than –1

Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

WARNINGS

Because in the future the variable **errno** is set to EAGAIN rather than EACCES when a section of a file is already locked by another process, portable application programs should expect and test for either value.

NAME

fork – create a new process

SYNOPSIS

#include <sys/types.h>
pid_t fork ()

DESCRIPTION

fork causes creation of a new process. The new process (child process) is an exact copy of the calling process (parent process). This means the child process inherits the following attributes from the parent process:

environment

close-on-exec flag (see **exec**(2))

signal handling settings (i.e., SIG_DFL , SIG_IGN , SIG_HOLD , function address)

set-user-ID mode bit

set-group-ID mode bit

profiling on/off status

nice value (see **nice**(2))

all attached shared memory segments (see **shmop**(2))

all attached mapped regions (see **mmap**(2seq))

process group ID

session ID

foreground process ID (see **exit**(2))

current working directory

root directory

file mode creation mask (see **umask**(2))

file size limit (see **ulimit**(2))

The child process differs from the parent process in the following ways:

The child process has a unique process ID.

The child process has a different parent process ID (i.e., the process ID of the parent process).

The child process has its own copy of the parent's file descriptors. Each of the child's file descriptors shares a common file pointer with the corresponding file descriptor of the parent.

All **semadj** values are cleared (see **semop**(2)).

Process locks, text locks and data locks are not inherited by the child (see **plock**(2)).

The child process's **utime**, **stime**, **cutime**, and **cstime** are set to 0. The time left until an alarm clock signal is reset to 0.

fork fails and no child process is created if one or more of the following are true:

[EAGAIN] The system-imposed limit on the total number of processes under execution would be exceeded.

[EAGAIN] The system-imposed limit on the total number of processes under execution by a single user would be exceeded.

[ENOMEM] There is insufficient swap space available for the process.

SEE ALSO

exec(2), nice(2), plock(2), ptrace(2), semop(2), shmop(2), signal(2), sigset(2), sigaction(2), times(2), ulimit(2), umask(2), wait(2)

DIAGNOSTICS

Upon successful completion, **fork** returns a value of 0 to the child process and returns the process ID of the child process to the parent process. Otherwise, a value of –1 is returned to the parent process, no child process is created, and **errno** is set to indicate the error.

NAME

getdents – read directory entries and put in a file system independent format

SYNOPSIS

```
#include <sys/dirent.h>

int getdents (fildes, buf, nbyte)
int fildes;
char *buf;
unsigned nbyte;
```

DESCRIPTION

fildes is a file descriptor obtained from an **open**(2) or **dup**(2) system call.

getdents attempts to read **nbyte** bytes from the directory associated with **fildes** and to format them as file system independent directory entries in the buffer pointed to by **buf**. Since the filesystem independent directory entries are of variable length, in most cases the actual number of bytes returned will be strictly less than **nbyte**.

The filesystem independent directory entry is specified by the **dirent** structure. For a description of this see **dirent**(4).

On devices capable of seeking, **getdents** starts at a position in the file given by the file pointer associated with **fildes**. Upon return from **getdents**, the file pointer is incremented to point to the next directory entry.

This system call was developed in order to implement the **readdir**(3X) routine [for a description see **directory**(3X)], and should not be used for other purposes.

getdents will fail if one or more of the following are true:

[EBADF]	**fildes** is not a valid file descriptor open for reading.
[EFAULT]	**buf** points outside the allocated address space.
[EINVAL]	**nbyte** is not large enough for one directory entry.
[EIO]	An I/O error occurred while accessing the filesystem.
[ENOENT]	The current file pointer for the directory is not located at a valid entry.
[ENOTDIR]	**fildes** is not a directory.

SEE ALSO

directory(3X), dirent(4)

DIAGNOSTICS

Upon successful completion a non-negative integer is returned indicating the number of bytes actually read. A value of 0 indicates the end of the directory has been reached. If the system call failed, a –1 is returned and **errno** is set to indicate the error.

NAME

getgroups – get supplementary group IDs

SYNOPSIS

```
#include <sys/types.h>

int getgroups (gidsetsize, grouplist)
int gidsetsize;
gid_t grouplist[];
```

DESCRIPTION

The **getgroups** function fills in the array grouplist with the supplementary group IDs of the calling process. The **gidsetsize** argument specifies the number of elements in the supplied array grouplist. The actual number of supplementary group IDs is returned. The values of array entries with indexes larger than or equal to the returned value are undefined.

It is unspecified whether the effective group ID of the calling process is included in or omitted from the returned list of supplementary group IDs.

As a special case, if the **gidsetsize** argument is zero, **getgroups** returns the number of supplementary group IDs associated with the calling process without modifying the array pointed to by the **grouplist** argument.

Upon successful competion, the number of supplementary group IDs is returned. This value is zero if {NGROUPS-MAX} is zero. A return value of -1 indicates failure and **errno** is set to indicate the error.

If the following condition occurs, **getgroups** returns -1 and sets **errno** to the corresponding value:

[EINVAL] The **gidsetsize** argument is not equal to zero and is less than the number of supplementary group IDs.

[EFAULT] The **grouplist** argument points outside the allocated address space for the process.

SEE ALSO

setuid(2)

NAME

getmsg – get next message off a stream

SYNOPSIS

```
#include <stropts.h>

int getmsg(fd, ctlptr, dataptr, flags)
int fd;
struct strbuf *ctlptr;
struct strbuf *dataptr;
int *flags;
```

DESCRIPTION

getmsg retrieves the contents of a message (see **intro**(2)) located at the stream head read queue from a STREAMS file, and places the contents into user specified buffer(s). The message must contain either a data part, a control part or both. The data and control parts of the message are placed into separate buffers, as described below. The semantics of each part is defined by the STREAMS module that generated the message.

fd specifies a file descriptor referencing an open stream. **ctlptr** and **dataptr** each point to a *strbuf* structure which contains the following members:

```
int maxlen;  /* maximum buffer length */
int len;     /* length of data        */
char *buf;   /* ptr to buffer     */
```

where **buf** points to a buffer in which the data or control information is to be placed, and **maxlen** indicates the maximum number of bytes this buffer can hold. On return, **len** contains the number of bytes of data or control information actually received, or is 0 if there is a zero-length control or data part, or is -1 if no data or control information is present in the message. **flags** may be set to the values 0 or RS_HIPRI and is used as described below.

ctlptr is used to hold the control part from the message and **dataptr** is used to hold the data part from the message. If **ctlptr** (or **dataptr**) is NULL or the **maxlen** field is -1, the control (or data) part of the message is not processed and is left on the stream head read queue and **len** is set to -1. If the **maxlen** field is set to 0 and there is a zero-length control (or data) part, that zero-length part is removed from the read queue and **len** is set to 0. If the **maxlen** field is set to 0 and there are more than zero bytes of control (or data) information, that information is left on the read queue and **len** is set to 0. If the **maxlen** field in **ctlptr** or **dataptr** is less than, respectively, the control or data part of the message, **maxlen** bytes are retrieved. In this case, the remainder of the message is left on the stream head read queue and a non-zero return value is provided, as described below under **DIAGNOSTICS**. If information is retrieved from a high-priority message, **flags** is set to RS_HIPRI on return.

By default, **getmsg** processes the first priority or non-priority message available on the stream head read queue. However, a user may choose to retrieve only priority messages by setting **flags** to RS_HIPRI. In this case, **getmsg** will only process the next message if it is a priority message.

If O_NDELAY has not been set, **getmsg** blocks until a message, of the type(s) specified by **flags** (priority or either), is available on the stream head read queue. If O_NDELAY has been set and a message of the specified type(s) is not present on the read queue, **getmsg** fails and sets **errno** to EAGAIN.

If a hangup occurs on the stream from which messages are to be retrieved, **getmsg** will continue to operate normally, as described above, until the stream head read queue is empty. Thereafter, it will return 0 in the **len** fields of **ctlptr** and **dataptr**.

getmsg fails if one or more of the following are true:

[EAGAIN] The O_NDELAY flag is set, and no messages are available.

[EBADF] **fd** is not a valid file descriptor open for reading.

[EBADMSG] Queued message to be read is not valid for **getmsg**.

[EFAULT] **ctlptr**, **dataptr**, or **flags** points to a location outside the allocated address space.

[EINTR] A signal was caught during the **getmsg** system call.

[EINVAL] An illegal value was specified in **flags**, or the stream referenced by **fd** is linked under a multiplexor.

[ENOSTR] A stream is not associated with **fd**.

A **getmsg** can also fail if a STREAMS error message had been received at the stream head before the call to **getmsg**. The error returned is the value contained in the STREAMS error message.

SEE ALSO

intro(2), read(2), poll(2), putmsg(2), write(2)

DIAGNOSTICS

Upon successful completion, a non-negative value is returned. A value of 0 indicates that a full message was read successfully. A return value of MORECTL indicates that more control information is waiting for retrieval. A return value of MOREDATA indicates that more data is waiting for retrieval. A return value of MORECTL | MOREDATA indicates that both types of information remain. Subsequent **getmsg** calls will retrieve the remainder of the message.

NAME

getpid, getpgrp, getppid – get process, process group, and parent process IDs

SYNOPSIS

#include <sys/types.h>

pid_t getpid ()

pid_t getpgrp ()

pid_t getppid ()

DESCRIPTION

getpid returns the process ID of the calling process.

getpgrp returns the process group ID of the calling process.

getppid returns the parent process ID of the calling process.

SEE ALSO

exec(2), fork(2), intro(2), setpgrp(2), setpgid(2), signal(2)

NAME

getuid, geteuid, getgid, getegid – get real user, effective user, real group, and effective group IDs

SYNOPSIS

```
#include <sys/types.h>

uid_t getuid ()

uid_t geteuid ()

gid_t getgid ()

gid_t getegid ()
```

DESCRIPTION

getuid returns the real user ID of the calling process.

geteuid returns the effective user ID of the calling process.

getgid returns the real group ID of the calling process.

getegid returns the effective group ID of the calling process.

SEE ALSO

intro(2), setuid(2)

NAME

ioctl – control device

SYNOPSIS

int ioctl (fildes, request, arg)
int fildes, request;

DESCRIPTION

ioctl performs a variety of control functions on devices and STREAMS. For non-STREAMS files, the functions performed by this call are *device-specific* control functions. The arguments **request** and **arg** are passed to the file designated by **fildes** and are interpreted by the device driver. This control is infrequently used on non-STREAMS devices, with the basic input/output functions performed through the **read**(2) and **write**(2) system calls.

For STREAMS files, specific functions are performed by the **ioctl** call as described in **streamio**(7).

fildes is an open file descriptor that refers to a device. **request** selects the control function to be performed and will depend on the device being addressed. **arg** represents additional information that is needed by this specific device to perform the requested function. The data type of **arg** depends upon the particular control request, but it is either an integer or a pointer to a device-specific data structure.

In addition to device-specific and STREAMS functions, generic functions are provided by more than one device driver, for example, the general terminal interface (see **termio**(7)).

ioctl fails for any type of file if one or more of the following are true:

[EBADF] **fildes** is not a valid open file descriptor.

[EFAULT] **request** requires a data transfer to or from a buffer pointed to by **arg**, but some part of the buffer is outside the process's allocated space.

[EINTR] A signal was caught during the **ioctl** system call.

ioctl also fails if the device driver detects an error. In this case, the error is passed through **ioctl** without change to the caller. A particular driver might not have all of the following error cases. Other requests to device drivers fail if any of the following are true:

[EINVAL] **request** or **arg** is not valid for this device.

[EIO] Some physical I/O error has occurred.

[ENOTTY] **fildes** is not associated with a device driver that accepts control functions.

[ENXIO] The **request** and **arg** are valid for this device driver, but the service requested can not be performed on this particular subdevice.

STREAMS errors are described in **streamio**(7).

SEE ALSO

streamio(7), termio(7)

DIAGNOSTICS

Upon successful completion, the value returned depends upon the device

control function, but must be a non-negative integer. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

kill – send a signal to a process or a group of processes

SYNOPSIS

int kill (pid, sig)
int pid, sig;

DESCRIPTION

kill sends a signal to a process or a group of processes. The process or group of processes to which the signal is to be sent is specified by **pid**. The signal that is to be sent is specified by **sig** and is either one from the list given in **signal**(2), or 0. If **sig** is 0 (the null signal), error checking is performed but no signal is actually sent. This can be used to check the validity of **pid**.

The real or effective user ID of the sending process must match the real or effective user ID of the receiving process, unless the effective user ID of the sending process is superuser.

Certain processes are exempt form system wide signals and are know as "System processes." These are:

init - system initialization process

sched - swapper daemon process

vhand - pageout daemon process

bdflush - disk block delayed flush daemon process

If **pid** is greater than zero, **sig** will be sent to the process whose process ID is equal to **pid**. **pid** may equal 1.

If **pid** is 0, **sig** will be sent to all processes excluding System Processes whose process group ID is equal to the process group ID of the sender.

If **pid** is –1 and the effective user ID of the sender is not superuser, **sig** will be sent to all processes excluding System Processes whose real user ID is equal to the effective user ID of the sender.

If **pid** is –1 and the effective user ID of the sender is superuser, **sig** will be sent to all processes excluding System Processes.

If **pid** is negative but not –1, **sig** will be sent to all processes whose process group ID is equal to the absolute value of **pid**.

kill will fail and no signal will be sent if one or more of the following are true:

[EINVAL] **Sig** is not a valid signal number.

[EINVAL] **sig** is SIGKILL and **pid** is 1 (**proc1**).

[EPERM] The user ID of the sending process is not superuser, and its real or effective user ID does not match the real or effective user ID of the receiving process.

[ESRCH] No process can be found corresponding to that specified by **pid**.

SEE ALSO

kill(1), getpid(2), setpgrp(2), signal(2), sigset(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

link – link to a file

SYNOPSIS

int link (path1, path2)
char *path1, *path2;

DESCRIPTION

path1 points to a pathname naming an existing file. **path2** points to a pathname naming the new directory entry to be created. **link** creates a new link (directory entry) for the existing file.

link will fail and no link is created if one or more of the following are true:

[EACCES]	A component of either path prefix denies search permission.
[EACCES]	The requested link requires writing in a directory with a mode that denies write permission.
[EEXIST]	The link named by **path2** exists.
[EFAULT]	**path1** or **path2** points outside the allocated address space of the process.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path1** or **path2.**
[EMLINK]	The maximum number of links to a file would be exceeded.
[ENAMETOOLONG]	**path1** or **path2** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path1** or **path2** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path1** or **path2** was longer than NAME_MAX characters.
[ENOENT]	A component of either path prefix does not exist.
[ENOENT]	The file named by **path1** does not exist.
[ENOENT]	**path2** points to a null pathname.
[ENOSPC]	The directory in which the entry for **path2** is being placed cannot be extended because there is no space left on the file system containing the directory.
[ENOTDIR]	A component of either path prefix is not a directory.
[EPERM]	The file named by **path1** is a directory and the effective user ID is not superuser.
[EROFS]	The requested link requires writing in a directory on a read-only filesystem.
[EXDEV]	The link named by **path2** and the file named by **path1** are on different logical devices (filesystems).

SEE ALSO

symlink(2), unlink(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error.

NAME

lseek – move read/write file pointer

SYNOPSIS

```
#include <sys/types.h>
#include <unistd.h>

off_t lseek (fildes, offset, whence)
int fildes;
off_t offset;
int whence;
```

DESCRIPTION

fildes is a file descriptor returned from a **creat, open, dup,** or **fcntl** system call. **lseek** sets the file pointer associated with **fildes** as follows:

If **whence** is SEEK_SET, the pointer is set to **offset** bytes.

If **whence** is SEEK_CUR, the pointer is set to its current location plus **offset.**

If **whence** is SEEK_END, the pointer is set to the size of the file plus **offset.**

Upon successful completion, the resulting pointer location, as measured in bytes from the beginning of the file, is returned.

Seeking far beyond the end of a file, then writing, creates a gap or "hole", which occupies no physical file system space and reads as zeros.

lseek fails and the file pointer is unchanged if any of the following are true:

[EBADF] **fildes** is not an open file descriptor.

[EINVAL and SIGSYS signal]
whence is not one of SEEK_SET, SEEK_CUR or SEEK_END.

[EINVAL] The resulting file pointer would be negative.

[ESPIPE] **fildes** is associated with a pipe or *fifo*.

Some devices are incapable of seeking. The value of the file pointer associated with such a device is undefined.

SEE ALSO

creat(2), dup(2), fcntl(2), open(2)

DIAGNOSTICS

Upon successful completion, a non-negative integer indicating the file pointer value is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

mkdir – make a directory

SYNOPSIS

```
#include <sys/types.h>
int mkdir (path, mode)
char *path;
mode_t mode;
```

DESCRIPTION

The routine **mkdir** creates a new directory with the name **path**. The mode of the new directory is initialized from the **mode**. The protection part of the **mode** argument is modified by the process's mode mask [see **umask**(2)].

The directory's owner ID is set to the process's effective user ID. The directory's group ID is set to the group ID of the directory in which this directory is being created. The newly created directory is empty with the possible exception of entries for "." and "..". **mkdir** fails and no directory is created if any of the following are true:

[EACCES]	Either a component of the path prefix denies search permission or write permission is denied on the parent directory of the directory to be created.
[EEXIST]	The named file already exists.
[EFAULT]	**path** points outside the allocated address space of the process.
[EIO]	An I/O error has occurred while accessing the filesystem.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[EMLINK]	The maximum number of links to the parent directory would be exceeded.
[ENAMETOOLONG]	The path is longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	A component of the path prefix does not exist.
[ENOSPC]	The directory in which the entry for the new directory is being placed cannot be expanded because there is no space left on the file system containing the directory.
[ENOSPC]	The new directory cannot be created because there is no space left on the file system which will contain the directory.
[ENOSPC]	There are no free inodes on the file system on which the directory is being created.

[ENOTDIR] A component of the path prefix is not a directory.

[EROFS] The path prefix resides on a read-only filesystem.

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of -1 is returned, and **errno** is set to indicate the error.

SEE ALSO

umask(2), chmod(2), stat(2), lstat(2)

NAME

mknod – make a directory, or a special or ordinary file

SYNOPSIS

```
int mknod (path, mode, dev)
        char *path;
        mode_t mode;
        dev_t dev
```

DESCRIPTION

mknod creates a new file named by the pathname pointed to by **path**. The mode of the new file is initialized from **mode**, where the value of **mode** is interpreted as follows:

0170000	file type; one of the following: 0010000 *fifo* special 0020000 character special 0040000 directory 0060000 block special 0100000 or 0000000 ordinary file
0004000	set user ID on execution
00020#0	set group ID on execution if # is 7, 5, 3, or 1 enable mandatory file/record locking if # is 6, 4, 2, or 0
0001000	save text image after execution
0000777	access permissions; constructed from the following: 0000400 read by owner 0000200 write by owner 0000100 execute (search on directory) by owner 0000070 read, write, execute (search) by group 0000007 read, write, execute (search) by others

The owner ID of the file is set to the effective user ID of the process. The group ID of the file is set to the group ID of the directory in which the file is being created.

Values of **mode** other than those above are undefined and should not be used. The low-order 9 bits of **mode** are modified by the process's file mode creation mask: all bits set in the process's file mode creation mask are cleared (see **umask**(2)). If **mode** indicates a block or character special file, **dev** is a configuration-dependent specification of a character or block I/O device. If **mode** does not indicate a block special or character special device, **dev** is ignored.

mknod may be invoked only by the superuser for file types other than FIFO special.

mknod fails and the new file is not created if any of the following are true:

[EACCESS]	A component of **path** denied search permission for the creation of a *fifo* by a non superuser.
[EEXIST]	The named file exists.
[EFAULT]	**path** points outside the allocated address space of the process.

[EIO]	An I/O error occurred while reading from or writing to the file system.
[EISDIR]	**path** is a directory.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	A component of the path prefix does not exist.
[ENOSPC]	The directory in which the entry for the new file is being placed cannot be expanded because there is no space left on the file system containing the directory.
[ENOSPC]	The new file cannot be created because there is no space left on the file system which will contain the file.
[ENOSPC]	There are no free inodes on the file system on which the file is being created.
[ENOTDIR]	A component of the path prefix is not a directory.
[EPERM]	The effective user ID of the process is not superuser.
[EROFS]	The directory in which the file is to be created is located on a read-only filesystem.

SEE ALSO

mkdir(1), chmod(2), exec(2), umask(2), fs(4)

DIAGNOSTICS

Upon successful completion a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

mount – mount a filesystem

SYNOPSIS

```
#include <sys/types.h>
#include <sys/mount.h>

int mount (spec, dir, mflag, fstyp, dataptr, datalen)
char *spec, *dir;
int mflag, fstyp;
char *dataptr;
int datalen;
```

DESCRIPTION

mount requests that a removable filesystem contained on the block special file identified by **spec** be mounted on the directory identified by **dir**. **spec** and **dir** are pointers to pathnames. **fstyp** is the filesystem type number. The filesystem type number can be determined from *<sys/mount.h>*. Note that if both the MS_DATA and MS_FSS flag bits of **mflag** are off, the filesystem type defaults to the root filesystem type. Only if either flag is on is **fstyp** used to indicate the filesystem type.

If the MS_DATA flag is set in **mflag** the system expects the **dataptr** and **datalen** arguments to be present. Together they describe a block of file-system specific data at address **dataptr** of length **datalen**. This is interpreted by file-system specific code within the operating system and its format depends upon the filesystem type. A particular filesystem type may not require this data, in which case **dataptr** and **datalen** should both be zero. Note that MS_FSS is obsolete and is ignored if MS_DATA is also set, but if MS_FSS is set and MS_DATA is not, **dataptr** and **datalen** are both assumed to be zero.

Upon successful completion, references to the file **dir** refer to the root directory on the mounted filesystem.

The low-order bit of **mflag** is used to control write permission on the mounted filesystem; if 1, writing is forbidden, otherwise writing is permitted according to individual file accessibility.

mount may be invoked only by the superuser. It is intended for use only by the **mount**(1M) utility.

mount fails if one or more of the following are true:

[EBUSY]	**dir** is currently mounted on, is someone's current working directory, or is otherwise busy.
[EBUSY]	The device associated with **spec** is currently mounted.
[EBUSY]	There are no more mount table entries.
[EFAULT]	**spec** or **dir** points outside the allocated address space of the process.
[EINVAL]	The super block has an invalid magic number or the **fstyp** is invalid or **mflag** is not valid.
[ELOOP]	Too many symbolic links were encountered in translating **spec** or **dir**.

[EMULTIHOP]	Components of **path** require hopping to multiple remote machines.
[ENAMETOOLONG]	**spec** or **dir** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**spec** or **dir** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **spec** or **dir** was longer than NAME_MAX characters.
[ENOENT]	Any of the named files does not exist.
[ENOLINK]	**path** points to a remote machine and the link to that machine is no longer active.
[ENOSPC]	The filesystem state in the superblock is not FsOKAY and **mflag** requests write permission.
[ENOTBLK]	**spec** is not a block special device.
[ENOTDIR]	A component of a path prefix is not a directory.
[ENOTDIR]	**dir** is not a directory.
[ENXIO]	The device associated with **spec** does not exist.
[EPERM]	The effective user ID is not superuser.
[EREMOTE]	**spec** is remote and cannot be mounted.
[EROFS]	**spec** is write protected and **mflag** requests write permission.

SEE ALSO

umount(2), mount(1M), fs(4)

DIAGNOSTICS

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error.

NAME

msgctl – message control operations

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgctl (msqid, cmd, buf)
int msqid, cmd;
struct msqid_ds *buf;
```

DESCRIPTION

msgctl provides a variety of message control operations as specified by **cmd**. The following **cmd**s are available:

IPC_STAT — Place the current value of each member of the data structure associated with **msqid** into the structure pointed to by **buf**. The contents of this structure are defined in **intro**(2). {READ}

IPC_SET — Set the value of the following members of the data structure associated with **msqid** to the corresponding value found in the structure pointed to by **buf**:

```
msg_perm.uid
msg_perm.gid
msg_perm.mode /* only low 9 bits */
msg_qbytes
```

This **cmd** can only be executed by a process that has an effective user ID equal to either that of super user, or to the value of **msg_perm.cuid** or **msg_perm.uid** in the data structure associated with **msqid**. Only super user can raise the value of **msg_qbytes**.

IPC_RMID — Remove the message queue identifier specified by **msqid** from the system and destroy the message queue and data structure associated with it. This **cmd** can only be executed by a process that has an effective user ID equal to either that of super user, or to the value of **msg_perm.cuid** or **msg_perm.uid** in the data structure associated with **msqid**.

msgctl fails if any of the following are true:

[EACCES] **cmd** is equal to IPC_STAT and {READ} operation permission is denied to the calling process (see **intro**(2)).

[EFAULT] **buf** points to an illegal address.

[EINVAL] **msqid** is not a valid message queue identifier.

[EINVAL] **cmd** is not a valid command.

[EPERM] **cmd** is equal to IPC_RMID or IPC_SET. The effective user ID of the calling process is not equal to that of super user, or to the value of **msg_perm.cuid** or **msg_perm.uid** in the data structure associated with **msqid**.

[EPERM] **cmd** is equal to IPC_SET, an attempt is being made to increase to the value of **msg_qbytes**, and the effective user ID of the calling

process is not equal to that of super user.

SEE ALSO

intro(2), msgget(2), msgop(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

msgget – get message queue

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgget (key, msgflg)
key_t key;
int msgflg;
```

DESCRIPTION

msgget returns the message queue identifier associated with **key**.

A message queue identifier and associated message queue and data structure (see **intro**(2)) are created for **key** if one of the following are true:

key is equal to IPC_PRIVATE .

key does not already have a message queue identifier associated with it, and (**msgflg** & IPC_CREAT) is true.

Upon creation, the data structure associated with the new message queue identifier is initialized as follows:

msg_perm.cuid, **msg_perm.uid**, **msg_perm.cgid**, and **msg_perm.gid** are set equal to the effective user ID and effective group ID, respectively, of the calling process.

The low-order 9 bits of **msg_perm.mode** are set equal to the low-order 9 bits of **msgflg**.

msg_qnum, **msg_lspid**, **msg_lrpid**, **msg_stime**, and **msg_rtime** are set equal to 0.

msg_ctime is set equal to the current time.

msg_qbytes is set equal to the system limit.

msgget fails if any of the following are true:

[EACCES] A message queue identifier exists for **key**, but operation permission (see **intro**(2)) as specified by the low-order 9 bits of **msgflg** would not be granted.

[EEXIST] A message queue identifier exists for **key** but ((**msgflg** & IPC_CREAT) & (**msgflg** & IPC_EXCL)) is true.

[ENOENT] A message queue identifier does not exist for **key** and (**msgflg** & IPC_CREAT) is false.

[ENOSPC] A message queue identifier is to be created but the system-imposed limit on the maximum number of allowed message queue identifiers system-wide would be exceeded.

SEE ALSO

intro(2), msgctl(2), msgop(2)

DIAGNOSTICS

Upon successful completion, a non-negative integer, namely a message queue identifier, is returned. Otherwise, –1 is returned and **errno** is set.

NAME

msgop – message operations

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgsnd (msqid, msgp, msgsz, msgflg)
int msqid;
struct msgbuf *msgp;
int msgsz, msgflg;

int msgrcv (msqid, msgp, msgsz, msgtyp, msgflg)
int msqid;
struct msgbuf *msgp;
int msgsz;
long msgtyp;
int msgflg;
```

DESCRIPTION

msgsnd is used to send a message to the queue associated with the message queue identifier specified by **msqid**. {WRITE} **msgp** points to a structure containing the message. This structure is composed of the following members:

```
long   mtype;     /* message type */
char   mtext[];   /* message text */
```

mtype is a positive integer that can be used by the receiving process for message selection (see **mtext** is any text of length **msgsz** bytes. **msgsz** can range from 0 to a system-imposed maximum.

msgflg specifies the action to be taken if one or more of the following are true:

The number of bytes already on the queue is equal to **msg_qbytes** [see **intro**(2)].

The total number of messages on all queues system-wide is equal to the system-imposed limit.

These actions are as follows:

If (**msgflg** & IPC_NOWAIT) is true, the message is not sent and the calling process returns immediately.

If (**msgflg** & IPC_NOWAIT) is false, the calling process suspends execution until one of the following occurs:

The condition responsible for the suspension no longer exists, in which case the message is sent.

msqid is removed from the system (see **msgctl**(2)). When this occurs, **errno** is set equal to EIDRM, and a value of –1 is returned.

The calling process receives a signal that is to be caught. In this case the message is not sent and the calling process resumes execution in the manner prescribed in **signal**(2).

msgsnd fails and no message is sent if one or more of the following are true:

[EACCES] Operation permission is denied to the calling process [see **intro**(2)].

[EAGAIN] The message cannot be sent for one of the reasons cited above and (**msgflg** & IPC_NOWAIT) is true.

[EFAULT] **msgp** points to an illegal address.

[EIDRM] The message queue associated with **msgquid** was deleted while the process was blocked in **msgsnd** waiting to send a message or while blocked in **msgrcv** waiting to receive a message.

[EINTR] A signal was caught during the **msgsnd** or **msgrcv** system call.

[EINVAL] **msqid** is not a valid message queue identifier.

[EINVAL] **mtype** is less than 1.

[EINVAL] **msgsz** is less than zero or greater than the system-imposed limit.

Upon successful completion, the following actions are taken with respect to the data structure associated with **msqid** (see **intro**(2)).

msg_qnum is incremented by 1.

msg_lspid is set equal to the process ID of the calling process.

msg_stime is set equal to the current time.

msgrcv reads a message from the queue associated with the message queue identifier specified by **msqid** and places it in the structure pointed to by **msgp**. {READ} This structure is composed of the following members:

```
long   mtype;     /* message type */
char   mtext[];   /* message text */
```

mtype is the received message's type as specified by the sending process. **mtext** is the text of the message. **msgsz** specifies the size in bytes of **mtext**. The received message is truncated to **msgsz** " bytes" if it is larger than **msgsz** and (**msgflg** & **MSG_NOERROR**) is true. The truncated part of the message is lost and no indication of the truncation is given to the calling process.

msgtyp specifies the type of message requested as follows:

If **msgtyp** is equal to 0, the first message on the queue is received.

If **msgtyp** is greater than 0, the first message of type **msgtyp** is received.

If **msgtyp** is less than 0, the first message of the lowest type that is less than or equal to the absolute value of **msgtyp** is received.

msgflg specifies the action to be taken if a message of the desired type is not on the queue. These are as follows:

If (**msgflg** & IPC_NOWAIT) is true, the calling process will return immediately with a return value of –1 and **errno** set to ENOMSG.

If (**msgflg** & IPC_NOWAIT) is false, the calling process will suspend execution until one of the following occurs:

A message of the desired type is placed on the queue.

msqid is removed from the system. When this occurs, **errno** is set equal to EIDRM, and a value of –1 is returned.

The calling process receives a signal that is to be caught. In this case a message is not received and the calling process resumes execution in the manner prescribed in *signal*(2).

msgrcv fails and no message is received if one or more of the following are true:

[EACCES] Operation permission is denied to the calling process.

[EFAULT] **msgp** points to an illegal address.

[EIDRM] The message queue associated with **msgquid** was deleted while the process was blocked in **msgsnd** waiting to send a message or while blocked in **msgrcv** waiting to receive a message.

[EINVAL] **msqid** is not a valid message queue identifier.

[EINVAL] **msgsz** is less than 0.

[E2BIG] **Mtext** is greater than **msgsz** and (**msgflg** & MSG_NOERROR) is false.

[ENOMSG] The queue does not contain a message of the desired type and (**msgtyp** & IPC_NOWAIT) is true.

Upon successful completion, the following actions are taken with respect to the data structure associated with **msqid** [see **intro**(2)].

msg_qnum is decremented by 1.

msg_lrpid is set equal to the process ID of the calling process.

msg_rtime is set equal to the current time.

SEE ALSO

intro(2), msgctl(2), msgget(2), signal(2)

DIAGNOSTICS

If **msgsnd** or **msgrcv** return due to the receipt of a signal, a value of –1 is returned to the calling process and **errno** is set to EINTR. If they return due to removal of **msqid** from the system, a value of –1 is returned and **errno** is set to EIDRM.

Upon successful completion, the return value is as follows:

msgsnd returns a value of 0.

msgrcv returns a value equal to the number of bytes actually placed into **mtext**.

Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

nice – change priority of a process

SYNOPSIS

int nice (incr)
int incr;

DESCRIPTION

nice adds the value of **incr** to the nice value of the calling process. A process's **nice** value is a non-negative number for which a more positive value results in lower CPU priority.

A maximum **nice** value of 39 and a minimum **nice** value of 0 are imposed by the system. (The default value is 20.) Requests for values above or below these limits result in the **nice** value being set to the corresponding limit.

[EPERM] **nice** fails and does not change the **nice** value if **incr** is negative or greater than 39 and the effective user ID of the calling process is not superuser.

SEE ALSO

nice(1), exec(2)

DIAGNOSTICS

Upon successful completion, **nice** returns the new nice value minus 20. Otherwise, a value of –1 is returned and *errno* is set to indicate the error.

NAME

open – open for reading or writing

SYNOPSIS

```
#include <sys/types.h>
#include <fcntl.h>
int open (path, oflag [, mode]
char *path;
int oflag;
mode_t mode;
```

DESCRIPTION

path points to a pathname naming a file. **open** opens a file descriptor for the named file and sets the file status flags according to the value of **oflag**. For non-STREAMS (see **intro**(2)) files, **oflag** values are constructed by ORing flags from the following list (only one of the first three flags below may be used):

O_RDONLY — Open for reading only.

O_WRONLY — Open for writing only.

O_RDWR — Open for reading and writing.

O_NONBLOCK — This flag may affect subsequent reads and writes (see **read**(2) and **write**(2)].

When opening a *FIFO* with O_RDONLY or O_WRONLY set:

If O_NONBLOCK is set:

An **open** for reading-only returns without delay. An **open** for writing-only returns an error if no process currently has the file open for reading.

If O_NONBLOCK is clear:

An **open** for reading-only blocks until a process opens the file for writing. An **open** for writing-only blocks until a process opens the file for reading.

When opening a file associated with a communication line:

If O_NONBLOCK is set:

The open returns without waiting for carrier.

If O_NONBLOCK is clear:

The open blocks until carrier is present.

O_NDELAY — Is a synonym for O_NONBLOCK.

O_APPEND — If set, the file pointer is set to the end of the file prior to each write.

O_SYNC — When opening a regular file, this flag affects subsequent writes. If set, each **write**(2) waits for both the file data and file status to be physically updated.

O_CREAT — If the file exists, this flag has no effect. Otherwise, the owner ID of the file is set to the effective user ID of the process, the group ID of the file is set to the group ID of the

directory in which this file is being created, and the low-order 12 bits of the file mode are set to the value of **mode** modified as follows (see **creat**(2)):

All bits set in the file mode creation mask of the process are cleared (see **umask**(2)).

The `save text image after execution bit` of the mode is cleared (see **chmod**(2)).

If the effective user ID of the process is not superuser and if the group ID of the file does not match the effective group ID of the process or one of the supplementary group IDs of the process, the mode bit `set group ID on execution`is cleared (see **chmod**(2)). Clearing this bit may also disable mandatory record locking on that file.

O_TRUNC — If the file exists, its length is truncated to 0 and the mode and owner are unchanged.

O_EXCL — If O_EXCL and O_CREAT are set, **open** fails if the file exists.

When opening a STREAMS file, **oflag** may be constructed from O_NONBLOCK OR-ed with either O_RDONLY, O_WRONLY or O_RDWR. Other flag values are not applicable to STREAMS devices and have no effect on them. The value of O_NONBLOCK affects the operation of STREAMS drivers and certain system calls (see **read**(2), **getmsg**(2), **putmsg**(2) and **write**(2)). For drivers, the implementation of O_NONBLOCK is device-specific. Each STREAMS device driver may treat this option differently.

Certain flag values can be set following **open** as described in **fcntl**(2).

The file pointer used to mark the current position within the file is set to the beginning of the file.

The new file descriptor is set to remain open across **exec** system calls (see **fcntl**(2)].

The named file is opened unless any of the following are true:

[EACCES] — A component of the path prefix denies search permission.

[EACCES] — **oflag** permission is denied for the named file.

[EACCES] — O_CREAT is specified, the file does not exist, and the directory in which the entry for the file is to be created does not permit writing.

[EAGAIN] — The file exists, mandatory file/record locking is set, and there are outstanding record locks on the file (see **chmod** (2)].

[EEXIST] — **O_CREAT** and **O_EXCL** are set, and the named file exists.

[EFAULT] — **path** points outside the allocated address space of the process.

[EINTR]	A signal was caught during the **open** system call.
[EIO]	A hangup or error occurred during a STREAMS **open**.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[EIO]	A special file generated an I/O error.
[EISDIR]	The named file is a directory and **oflag** is write or read/write.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[EMFILE]	**NOFILES** file descriptors are currently open (see **intro**(2)).
[ENAMETOOLONG]	**Path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**Path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENFILE]	The system file table is full.
[ENOENT]	**O_CREAT** is not set and the named file does not exist.
[ENOENT]	A component of **path** which must exist does not exist.
[ENOSPC]	O_CREAT is set, and the filesystem is out of inodes.
[ENOSPC]	O_CREAT is specified, and the directory in which the entry for the new file is being placed cannot be expanded because there is no space left on the file system containing the directory.
[ENOSR]	Unable to allocate a *stream*.
[ENOTDIR]	A component of the path prefix is not a directory.
[ENXIO]	The named file is a character special or block special file, and the device associated with this special file does not exist.
[ENXIO]	O_NONBLOCK is set, the named file is a *FIFO,* O_WRONLY is set, and no process has the file open for reading.
[ENXIO]	A STREAMS module or driver open routine failed.
[EROFS]	The named file resides on a read-only filesystem and **oflag** is write or read/write.
[ETXTBSY]	The file is a pure procedure (shared text) file that is being executed and **oflag** is write or read/write.

SEE ALSO

chmod(2), close(2), creat(2), dup(2), fcntl(2), intro(2), lseek(2), read(2), getmsg(2), putmsg(2), umask(2), write(2), getdtablesize(2seq)

DIAGNOSTICS

Upon successful completion, the file descriptor is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error.

NAME

pathconf, fpathconf – get configurable pathname variables

SYNOPSIS

```
include <unistd.h>

long pathconf (path, name)
char *path;
int name;

long fpathconf (fildes, name)
int fildes, name;
```

DESCRIPTION

The **pathconf** and **fpathconf** functions provide a method for the application to determine the current value of a configurable limit or option (variable) that is associated with a file or a directory.

For **pathconf**, the **path** argument points to the pathname of a file or directory.

For **fpathconf**, the **fildes** argument is an open file descriptor.

The **name** argument represents the variable to be queried relative to that file or directory.

The following table lists the supported variables. These variables come from *<limits.h>* or *<unistd.h>* and the symbolic constants, defined in *<unistd.h>*, that are the corresponding values used for **name:**

Configurable Pathname Variables

Variable	*name* **Value**	**Notes**
LINK_MAX	_PC_LINK_MAX	1
MAX_CANON	_PC_MAX_CANON	2
MAX_INPUT	_PC_MAX_INPUT	2
NAME_MAX	_PC_NAME_MAX	3,4
PATH_MAX	_PC_PATH_MAX	4,5
PIPE_BUF	_PC_PIPE_BUF	6
_POSIX_CHOWN_RESTRICTED	_PC_CHOWN_RESTRICTED	7
_POSIX_NO_TRUNG	_PC_NO_TRUNG	2
_POSIX_VDISABLE	_PC_VDISABLE	2

The following notes apply to the entries above:

1. If **path** or **fildes** refers to a directory, the value returned applies to the directory itself.

2. The behavior is undefined if **path** or **fildes** does not refer to a terminal file.

3. If **path** or **fildes** refers to a directory, the value returned applies to filenames within the directory.

4. The behavior is undefined if **path** or **fildes** does not refer to a directory.

5. If **path** or **fildes** refers to a directory, the value returned is the maximum length of a relative pathname when the specified directory is the working directory.

6. If **path** refers to a FIFO, or **fildes** refers to a pipe or a FIFO, the value returned applies to the referenced object itself. If **path** or **fildes** refer to a directory, the value returned applies to any FIFOs that exist or that can be created within the directory. If **path** or **fildes** refer to any other type of file, the behavior is undefined.

7. If **path** or **fildes** refer to a directory, the value returned applies to any files defined in this standard, other than directories, that exist or that can be created within the directory.

DIAGNOSTICS

If **name** is an invalid value, **pathconf** and **fpathconf** return -1.

If the variable corresponding to **name** has no limit for the path or file descriptor, **pathconf** and **fpathconf** return -1 without changing **errno.**

If **path** is needed to determine the value of *name*, and the association of **name** with the file specified by *path* is not supported, or if the process did not have the appropriate privileges to query the file specified by *path*, or **path** does not exist, the **pathconf** function returns -1.

If **fildes** is needed to determine the value of *name*, and the association of name with the file specfied by *fildes* is not supported, or if **fildes** is an invalid file descriptor, **fpathconf** returns -1.

Otherwise, **pathconf** and **fpathconf** return the current variable value for the file or directory without changing **errno.** The value returned is not more restrictive than the corresponding value described to the application when it was compiled with *<limits.h>* or *<unistd.h>*.

If the following condition occurs, **pathconf** and **fpathconf** return -1 and set **errno** to the corresponding value:

[EINVAL]	The value of name is invalid.

If any of the following conditions is detected, **pathconf** returns -1 and sets **errno** to the corresponding value:

[EACCES]	Search permmision is denied for a component of the path prefix.
[EINVAL]	The association of the variable name with the specified file is not supported.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	The length of the path argument exceeds PATH_MAX, or a pathname component is longer than NAME_MAX while _POSIX_NO_TRUNC is in effect.
[ENAMETOOLONG]	**Path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.

[ENOENT]	The named file does not exist or the path argument points to an empty string.
[ENOTDIR]	A component of the path prefix is not a directory.
[EFAULT]	**path** points outside the allocated address space of the process.

If any of the following conditions is detected, **fpathconf** returns -1 and sets **errno** to the corresponding value:

[EBADF]	The **fildes** argument is not a valid file descriptor.
[EINVAL]	The association of the variable name with the specified file is not supported.

NAME

pause – suspend process until signal

SYNOPSIS

pause ()

DESCRIPTION

pause suspends the calling process until it receives a signal. The signal must be one that is not currently set to be ignored by the calling process.

If the signal causes termination of the calling process, **pause** will not return.

If the signal is caught by the calling process and control is returned from the signal-catching function (see **signal**(2)), the calling process resumes execution from the point of suspension; with a return value of –1 from **pause** and **errno** set to EINTR.

SEE ALSO

alarm(2), kill(2), signal(2), sigpause(2), wait(2)

NAME

pipe – create an interprocess channel

SYNOPSIS

int pipe (fildes)
int fildes[2];

DESCRIPTION

pipe creates an I/O mechanism called a pipe and returns two file descriptors, **fildes**[0] and **fildes**[1]. **fildes**[0] is opened for reading and **fildes**[1] is opened for writing.

Up to PIPE_BUF bytes of data are buffered by the pipe before the writing process is blocked. A read-only file descriptor **fildes**[0] accesses the data written to **fildes**[1] on a first-in-first-out (FIFO) basis.

pipe fails if any of the following are true:

[EMFILE]	NOFILES file descriptors are currently open (see **intro**(2)).
[ENFILE]	The system file table is full.
[ENFILE]	The number of pipes in the system exceeds the configured maxinum number of pipes.

If **fildes** is an invalid address in the process's address space, the results will be unpredictable, possibly resulting in a core dump.

SEE ALSO

sh(1), intro(2), read(2), write(2), getdtablesize(2seq)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

plock – lock process, text, or data in memory

SYNOPSIS

```
#include <sys/lock.h>

int plock (op)
int op;
```

DESCRIPTION

plock allows the calling process to lock its text segment (text lock), its data segment (data lock), or both its text and data segments (process lock) into memory. Locked segments are immune to all routine swapping. **plock** also allows these segments to be unlocked. The effective user ID of the calling process must be superuser to use this call. **op** specifies the following:

PROCLOCK	lock text and data segments into memory (process lock)
TXTLOCK	lock text segment into memory (text lock)
DATLOCK	lock data segment into memory (data lock)
UNLOCK	remove locks

plock fails and does not perform the requested operation if any of the following are true:

[EAGAIN]	Not enough memory.
[EINVAL]	**op** is equal to **PROCLOCK** and a process lock, a text lock, or a data lock already exists on the calling process.
[EINVAL]	**op** is equal to **TXTLOCK** and a text lock, or a process lock already exists on the calling process.
[EINVAL]	**op** is equal to **DATLOCK** and a data lock, or a process lock already exists on the calling process.
[EINVAL]	**op** is equal to **UNLOCK** and no type of lock exists on the calling process.
[EPERM]	The effective user ID of the calling process is not superuser.

SEE ALSO

exec(2), exit(2), fork(2), vm_ctl(2seq)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned to the calling process. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NOTES

Currently, the entire process is marked as being immune to all swapping when any of the individual segments are locked. Programs should not mix **plock** calls with the VM_SWAPABLE case of **vm_ctl**(2seq) since they have the same effect and can interact with each other. In particular, a call to **vm_ctl** followed by calls to **plock** to lock and subsequently unlock the process will have the effect of disabling the original call to **vm_ctl.**

NAME

poll – STREAMS input/output multiplexing

SYNOPSIS

```
#include <stropts.h>
#include <poll.h>

int poll(fds, nfds, timeout)
struct pollfd fds[];
unsigned long nfds;
int timeout;
```

DESCRIPTION

poll provides users with a mechanism for multiplexing input/output over a set of file descriptors that reference open streams (see **intro**(2)). **poll** identifies those streams on which a user can send or receive messages, or on which certain events have occurred. A user can receive messages using **read**(2) or **getmsg**(2) and can send messages using **write**(2) and **putmsg**(2). Certain **ioctl**(2) calls, such as I_RECVFD and I_SENDFD (see **streamio**(7)), can also be used to receive and send messages.

fds specifies the file descriptors to be examined and the events of interest for each file descriptor. It is a pointer to an array with one element for each open file descriptor of interest. The array's elements are **pollfd** structures which contain the following members:

```
int fd;                 /* file descriptor */
short events;           /* requested events */
short revents;          /* returned events */
```

where **fd** specifies an open file descriptor and **events** and **revents** are bitmasks constructed by or-ing any combination of the following event flags:

POLLIN — A non-priority or file descriptor passing message (see I_RECVFD) is present on the stream head read queue. This flag is set even if the message is of zero length. In **revents**, this flag is mutually exclusive with POLLPRI.

POLLPRI — A priority message is present on the stream head read queue. This flag is set even if the message is of zero length. In **revents**, this flag is mutually exclusive with POLLIN.

POLLOUT — The first downstream write queue in the stream is not full. Priority control messages can be sent (see **putmsg**) at any time.

POLLERR — An error message has arrived at the stream head. This flag is only valid in the **revents** bitmask; it is not used in the **events** field.

POLLHUP — A hangup has occurred on the stream. This event and POLLOUT are mutually exclusive; a stream can never be writable if a hangup has occurred. However, this event and POLLIN or POLLPRI are not mutually exclusive. This flag is only valid in the **revents** bitmask; it is not used in the **events** field.

POLLNVAL The specified **fd** value does not belong to an open stream. This flag is only valid in the **revents** field; it is not used in the **events** field.

For each element of the array pointed to by **fds**, **poll** examines the given file descriptor for the event(s) specified in **events**. The number of file descriptors to be examined is specified by **nfds**. If **nfds** exceeds NOFILES, the system limit of open files (see **intro**(2)), **poll** fails.

If the value **fd** is less than zero, **events** is ignored and **revents** is set to 0 in that entry on return from **poll**.

The results of the **poll** query are stored in the **revents** field in the **pollfd** structure. Bits are set in the **revents** bitmask to indicate which of the requested events are true. If none are true, none of the specified bits is set in **revents** when the **poll** call returns. The event flags POLLHUP, POLLERR and POLLNVAL are always set in **revents** if the conditions they indicate are true; this occurs even though these flags were not present in **events**.

If none of the defined events have occurred on any selected file descriptor, **poll** waits at least **timeout** msec for an event to occur on any of the selected file descriptors. On a computer where millisecond timing accuracy is not available, **timeout** is rounded up to the nearest legal value available on that system. If the value **timeout** is 0, **poll** returns immediately. If the value of **timeout** is -1, **poll** blocks until a requested event occurs or until the call is interrupted. **poll** is not affected by the O_NDELAY flag.

poll fails if one or more of the following are true:

[EAGAIN] Allocation of internal data structures failed but request should be attempted again.

[EFAULT] Some argument points outside the allocated address space.

[EINTR] A signal was caught during the **poll** system call.

[EINVAL] The argument **nfds** is less than zero, or **nfds** is greater than NOFILES (see **intro**(2)).

SEE ALSO

intro(2), getmsg(2), putmsg(2), read(2), streamio(7), write(2)

DIAGNOSTICS

Upon successful completion, a non-negative value is returned. A positive value indicates the total number of file descriptors that has been selected (i.e., file descriptors for which the **revents** field is non-zero). A value of 0 indicates that the call timed out and no file descriptors have been selected. Upon failure, a value of -1 is returned and **errno** is set to indicate the error.

NAME

profil – execution time profile

SYNOPSIS

void profil (buff, bufsiz, offset, scale)
char *buff;
int bufsiz, offset, scale;

DESCRIPTION

buff points to an area of core whose length (in bytes) is given by **bufsiz**. After this call, the user's program counter (pc) is examined each clock tick. Then the value of **offset** is subtracted from it, and the remainder multiplied by **scale**. If the resulting number corresponds to an entry inside **buff**, that entry is incremented. An entry is defined as a series of bytes with length **sizeof(short)**.

The scale is interpreted as an unsigned, fixed-point fraction with binary point at the left: 0177777 (octal) gives a 1-1 mapping of pc's to entries in **buff**; 077777 (octal) maps each pair of instruction entries together. 02(octal) maps all instructions onto the beginning of **buff** (producing a non-interrupting core clock).

Profiling is turned off by giving a **scale** of 0 or 1. It is rendered ineffective by giving a **bufsiz** of 0. Profiling is turned off when an **exec** is executed, but remains on in child and parent both after a **fork**. Profiling will be turned off if an update in **buff** would cause a memory fault.

SEE ALSO

prof(1), times(2), monitor(3C)

DIAGNOSTICS

Not defined.

NAME

ptrace, mptrace – process trace facility

SYNOPSIS

```
#include <signal.h>
#include <sys/ptrace.h>

ptrace(request, pid, addr, data)
int request, pid, *addr, data;

mptrace(request, pid, addr, data)
int request, pid, *addr, data;
```

DESCRIPTION

ptrace and **mptrace** provide a means by which a process may control the execution of other processes, and examine and change their memory image. Their primary use is for the implementation of breakpoint debugging.

There are four arguments whose interpretation depends on a **request** argument. Generally, **pid** is the process ID of the traced process, which is typically a descendant of the tracing process. A process being traced behaves normally until it encounters some signal, either internally generated like *illegal instruction* or externally generated like *interrupt* (see **signal**(2) for the list), or performs one of several state changes (**fork, exec, exit**). Then the traced process enters a stopped state and its debugger is notified with SIGCHLD. In the case of a fork, only the child stops (to allow the debugger to maintain a process tree model). When a process is in the stopped state, its memory image can be examined and modified using **ptrace** or **mptrace**. If desired, another **ptrace** or **mptrace** request can then cause the process either to terminate or to continue, possibly taking a signal.

The value of the **request** argument determines the precise action of the call, for **ptrace:**

PT_CHILD

This request is the only one used by the child process; it declares that the process is to be traced by its parent, which is not using any of the XPT_ functions. All the other arguments are ignored. Peculiar results will ensue if the parent does not expect to trace the child. The child process requesting to be traced must **exec(2)** an image before **ptrace** requests generated by the parent process are effective. PT_CHILD is not used by multi-process debuggers.

PT_RDATA, PT_RTEXT

The word in the process's address space at **addr** is returned as a function return value. If I and D space are separated (e.g., historically on a pdp-11), request PT_RTEXT indicates I space, PT_RDATA D space. **addr** must be even. The process must be stopped. The input **data** is ignored.

PT_RUSER

The word of the system's per-process data area corresponding to **addr** is returned as a function return value. **addr** must be even and less than *sizeof(struct user)* or correspond to the actual kernel address. This space contains the registers and other information about the process; its layout corresponds to the **user** structure in the system (see

<sys/user.h>). The value of the **data** argument is ignored.

PT_WDATA, PT_WTEXT
: The given **data** is written at the word in the process's address space corresponding to **addr.** No useful value is returned.

PT_WUSER
: The process's system **data** is written, as it is read with request PT_RUSER. Only a few locations can be written in this way: the general registers, the floating-point status and registers, and certain bits of the processor status word.

PT_CONTSIG
: The *data* argument is taken as a signal number and the process's execution continues at location **addr** as if it had incurred that signal. Normally the signal number will be either 0 to indicate that the signal that caused the stop should be ignored, or that value fetched out of the process's image indicating which signal caused the stop. If **addr** is `(int *)1` then execution continues from where it stopped.

PT_KILL
: The traced process terminates. However, if the process is a descendent of a multi-process debugger, the process will stop in exit (if it has not already done so), waiting to be continued before proceeding to terminate.

PT_SSTEP
: Execution continues as in request PT_CONTSIG; however, as soon as possible after execution of at least one instruction, execution stops again. The signal number from the stop is SIGTRAP. This is part of the mechanism for implementing breakpoints.

Unlike a single process debugger all decendants of the process declared as a multi-process debugger are traced and do not require themselves to use PT_CHILD as in ptrace(2). These are the arguments to **mptrace**:

XPT_DEBUGGER
: A process uses this request to declare itself a multi-process debugger. All other arguments are ignored. All subsequent **forks**, **execs**, and **exits**, as well as signals of all children of the calling process, their descendants, and selected other processes (see XPT_DEBUG), notify the debugger with SIGCHLD.

XPT_SETSIGMASK
: The **addr** parameter points to a mask of signals in the selected process. The mask may be set using **sigaddset**() or cleared using **sigdelset**(). Signals so set are passed directly to the selected process, rather than notifying the debugger and stopping the process. This mask is inherited across forks of the process, and kept when the process **exec**'s. A passed-through signal is a process attribute; it is passed through regardless of how many debuggers are controlling the process.

XPT_RREGS, XPT_WREGS
: The selected process's register set is read or written. **addr** points to a **pt_regset** structure (see *ptrace.h*). Machine specific restrictions

usually apply; for example, typically only a few of the processor-status register bits are actually written.

XPT_RPROC
: The selected process's process table entry is placed at ***addr.** See *<sys/proc.h>* for definition of `struct proc`.

XPT_SIGNAL
: The **data** argument is a signal number that is sent to process selected by **pid.** If **pid==0,** the signal is sent to all processes under control of the debugger.

XPT_STOPSTAT
: Information about a stopped process is returned. The **pid** argument is ignored. A debugged process stops whenever it **forks, execs, exits,** or is about to process a signal, and sends the debugger a SIGCHLD. **addr** points at a **pt_stop** structure, which receives the **pid** and reason for the stop. If no controlled process is stopped that has not yet been reported by XPT_STOPSTAT, an ESRCH error is returned.

XPT_DUPFD
: The open file descriptor in the caller given by **data** is **dup**'d (see **dup**(2)) into **pid's** file-descriptor named by **addr.** If **pid** already has an open file-descriptor there it is closed. This is an experimental interface.

The following functions are not supported in the first implementation:

XPT_DEBUG
: The process tree rooted at **pid** becomes controlled by the debugger. **pid** must not be an ancestor of the debugger or already under debug. The caller must be root or match the effective **uid** of all processes in the selected process tree. No process in the tree can be running a set-user or set-group process.

XPT_OPENT
: ptrace returns an open, read-only file descriptor for the binary file being executed by **pid.** The debugger must have read-access to the file for this to succeed. This allows the debugger to access the symbol table without having to know the name of the executable file.

To forestall possible fraud, **ptrace** inhibits the set-user-id and set-group-id facilities on subsequent **execve**(2) calls.

DIAGNOSTICS

A 0 value is returned if the call succeeds. If the call fails, a –1 is returned and **errno** is set to indicate the error. However, some ptrace commands return data as the function return value. Since –1 may be a legal (successful) return value, **errno** should be used to disambiguate.

The following errors may occur:

[EBADF] The data or addr argument specifies a bad file descriptor number for XPT_DUPFD.

[EFAULT] The specified address is out of bounds.

[EINVAL] The given signal number is invalid.

[EINVAL]	The request code is invalid.
[EIO]	The child cannot perform the task requested.
[EPERM]	The specified process cannot be traced.
[ESRCH]	The specified process does not exist.
[ESRCH]	The process is not stopped.
[ESRCH]	The process to be traced is not a descendant of the current process.

SEE ALSO

wait(2), signal(2)

BUGS

The –1 error indication is a legitimate function value; **errno**, can be used to disambiguate. (see **intro**(2))

Physical mapped address space, (MM_PHYS, can't be read or written using these interfaces (see mmap(2)).

NOTES

To use **mptrace**, *libseq* must be linked to the image using the **–lseq** option to the loader. The **mptrace**-specific structures mentioned above are defined below; the **pt_regset** structure is machine dependent.

```
structpt_stop      {
int    ps_pid;             /* pid of stopped process */
int    ps_reason;   /* reason for the stop */
};

structpt_regset {
/*
* General registers and processor status.
*/
int    pr_r0;
int    pr_r1;
int    pr_r2;
int    pr_r3;
int    pr_r4;
int    pr_r5;
int    pr_r6;
int    pr_r7;
int    pr_fp;
int    pr_sp;
int    pr_pc;
int    pr_modpsr;
/*
* Floating point registers and status.
*/
int    pr_f0;
int    pr_f1;
int    pr_f2;
int    pr_f3;
```

```
int   pr_f4;
int   pr_f5;
int   pr_f6;
int   pr_f7;
int   pr_fsr;
};
```

Reasons for a stop are a signal number (1 - NSIG), or one of PTS_FORK, PTS_EXEC, or PTS_EXIT, encoded to be unambiguous. A PTS_FORK stop only occurs in the child of the fork. An immediate child of the debugger does not stop with PTS_FORK.

NAME

putmsg – send a message on a stream

SYNOPSIS

```
#include <stropts.h>

int putmsg (fd, ctlptr, dataptr, flags)
int fd;
struct strbuf *ctlptr;
struct strbuf *dataptr;
int flags;
```

DESCRIPTION

putmsg creates a message (see **intro**(2)) from user-specified buffer(s) and sends the message to a STREAMS file. The message may contain either a data part, a control part or both. The data and control parts to be sent are distinguished by placement in separate buffers, as described below. The semantics of each part are defined by the STREAMS module that receives the message.

fd specifies a file descriptor referencing an open stream. **ctlptr** and **dataptr** each point to a **strbuf** structure which contains the following members:

```
int maxlen;  /* not used  */
int len;     /* length of data  */
char *buf;   /* ptr to buffer  */
```

ctlptr points to the structure describing the control part, if any, to be included in the message. The **buf** field in the **strbuf** structure points to the buffer where the control information resides, and the **len** field indicates the number of bytes to be sent. The **maxlen** field is not used in **putmsg** (see **getmsg**(2)). In a similar manner, **dataptr** specifies the data, if any, to be included in the message. **flags** may be set to the values 0 or RS_HIPRI and is used as described below.

To send the data part of a message, **dataptr** must be non-NULL and the **len** field of **dataptr** must have a value of 0 or greater. To send the control part of a message, the corresponding values must be set for **ctlptr**. No data (control) part is sent if either **dataptr** (**ctlptr**) is NULL or the **len** field of **dataptr** (**ctlptr**) is set to -1.

If a control part is specified, and **flags** is set to RS_HIPRI, a high-priority message is sent. If **flags** is set to 0, a non-priority message is sent. If no control part is specified, and **flags** is set to RS_HIPRI, **putmsg** fails and sets **errno** to EINVAL. If no control part and no data part are specified, and **flags** is set to 0, no message is sent, and 0 is returned.

For non-priority messages, **putmsg** blocks if the stream write queue is full due to internal flow control conditions. **putmsg** does not block for priority messages. For non-priority messages, **putmsg** does not block when the write queue is full and O_NDELAY is set. Instead, it fails and sets **errno** to EAGAIN.

putmsg also blocks, unless prevented by lack of internal resources, waiting for the availability of message blocks in the stream, regardless of priority or whether O_NDELAY has been specified. No partial message is sent.

putmsg fails if any of the following are true:

[EAGAIN] A non-priority message was specified, the **O_NDELAY** flag is set and the stream write queue is full due to internal flow control conditions.

[EAGAIN] Buffers could not be allocated for the message that was to be created.

[EBADF] **fd** is not a valid file descriptor open for writing.

[EFAULT] **ctlptr** or **dataptr** points outside the allocated address space.

[EINTR] A signal was caught during the **putmsg** system call.

[EINVAL] An undefined value was specified in **flags**, or **flags** is set to RS_HIPRI and no control part was supplied.

[EINVAL] The stream referenced by **fd** is linked below a multiplexor.

[ENOSTR] A stream is not associated with **fd**.

[ENXIO] A hangup condition was generated downstream for the specified stream.

[ERANGE] The size of the data part of the message does not fall within the range specified by the maximum and minimum packet sizes of the topmost stream module. This value is also returned if the control part of the message is larger than the maximum configured size of the control part of a message, or if the data part of a message is larger than the maximum configured size of the data part of a message.

A **putmsg** also fails if a STREAMS error message had been processed by the stream head before the call to **putmsg**. The error returned is the value contained in the STREAMS error message.

SEE ALSO

intro(2), read(2), getmsg(2), poll(2), write(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of -1 is returned and **errno** is set to indicate the error.

NAME

read – read from file

SYNOPSIS

int read (fildes, buf, nbyte)
int fildes;
char *buf;
unsigned nbyte;

DESCRIPTION

fildes is a file descriptor obtained from a **creat**(2), **open**(2), **dup**(2), **fcntl**(2), or **pipe**(2) system call.

read attempts to read **nbyte** bytes from the file associated with **fildes** into the buffer pointed to by **buf**.

On devices capable of seeking, the **read** starts at a position in the file given by the file pointer associated with **fildes**. Upon return from **read**, the file pointer is incremented by the number of bytes actually read.

Devices that are incapable of seeking always read from the current position. The value of a file pointer associated with such a file is undefined.

Upon successful completion, **read** returns the number of bytes actually read and placed in the buffer; this number may be less than **nbyte** if the file is associated with a communication line (see **ioctl**(2) and **termio**(7)), or if the number of bytes left in the file is less than **nbyte** bytes. A value of 0 is returned when an end-of-file has been reached.

A **read** from a STREAMS file (see **intro**(2)) can operate in three different modes: *byte-stream mode* , *message-nondiscard mode*, and *message-discard mode*. The default is *byte-stream mode*. This can be changed using the I_SRDOPT **ioctl** request (see **streamio**(7)), and can be tested with the I_GRDOPT **ioctl**. In byte-stream mode, **read** retrieves data from the **stream** until it has retrieved **nbyte** bytes, or until there is no more data to be retrieved. Byte-stream mode ignores message boundaries.

In STREAMS message-nondiscard mode, **read** retrieves data until it has read **nbyte** bytes, or until it reaches a message boundary. If the **read** does not retrieve all the data in a message, the remaining data are replaced on the **stream**, and can be retrieved by the next **read** or **getmsg**(2) call. Message-discard mode also retrieves data until it has retrieved **nbyte** bytes, or it reaches a message boundary. However, unread data remaining in a message after the **read** returns are discarded, and are not available for a subsequent **read** or **getmsg**.

When attempting to read from a regular file with mandatory file/record locking set (see **chmod**(2)), and there is a blocking (i.e. owned by another process) write lock on the segment of the file to be read:

If O_NONBLOCK is set, the read returns a -1 and sets **errno** to EAGAIN.

If O_NONBLOCK is clear, the read sleeps until the blocking record lock is removed.

When attempting to read from an empty pipe (or FIFO):

If O_NONBLOCK is set, the read returns a -1 and sets **errno** to EAGAIN.

If O_NONBLOCK is clear, the read blocks until data is written to the file or the file is no longer open for writing.

When attempting to read a file associated with a *tty* that has no data currently available:

If O_NONBLOCK is set, the read returns a -1 and sets **errno** to EAGAIN.

If O_NONBLOCK is clear, the read blocks until data becomes available.

When attempting to read a file associated with a stream that has no data currently available:

If O_NONBLOCK is set, the read returns a -1 and sets **errno** to EAGAIN.

If O_NONBLOCK is clear, the read blocks until data becomes available.

When reading from a STREAMS file, handling of zero-byte messages is determined by the current read mode setting. In byte-stream mode, **read** accepts data until it has read **nbyte** bytes, or until there is no more data to read, or until a zero-byte message block is encountered. **read** then returns the number of bytes read, and places the zero-byte message back on the **stream** to be retrieved by the next **read** or **getmsg**. In the two other modes, a zero-byte message returns a value of 0 and the message is removed from the **stream**. When a zero-byte message is read as the first message on a stream, a value of 0 is returned regardless of the read mode.

A **read** from a STREAMS file can only process data messages. It cannot process any type of protocol message and fails if a protocol message is encountered at the stream head.

read fails if any of the following are true:

[EAGAIN]	Mandatory file/record locking was set, O_NONBLOCK was set, and there was a blocking record lock.
[EAGAIN]	Total amount of system memory available when reading via raw I/O is temporarily insufficient.
[EAGAIN]	No message waiting to be read on a stream and O_NONBLOCK flag set.
[EBADF]	**fildes** is not a valid file descriptor open for reading.
[EBADMSG]	Message waiting to be read on a stream is not a data message.
[EDEADLK]	The read was going to go to sleep, causing a deadlock situation to occur.
[EFAULT]	**buf** points outside the allocated address space.
[EINTR]	A signal was caught during the **read** system call.
[EINVAL]	Attempted to read from a stream linked to a multiplexor.
[EIO]	An I/O error occurred while reading from or writing to the file system.

[EIO] The process is in a background process group and is attempting to read from its controlling terminal. Either the process is ignoring or blocking the SIGTTIN signal, or the process group of the process is orphaned.

[ENOLCK] The system record lock table was full, so the read could not go to sleep until the blocking record lock was removed.

[ENOMEM] The size of the read requested from a raw device would have required the process to exceed its maximum resident set size.

A **read** from a STREAMS file also fails if an error message is received at the stream head. In this case, **errno** is set to the value returned in the error message. If a hangup occurs on the stream being read, **read** continues to operate normally until the stream head read queue is empty. Thereafter, it returns 0.

SEE ALSO

creat(2), dup(2), fcntl(2), ioctl(2),intro(2), open(2), pipe(2), getmsg(2), streamio(7), termio(7)

DIAGNOSTICS

Upon successful completion a non-negative integer is returned indicating the number of bytes actually read. Otherwise, a –1 is returned and **errno** is set to indicate the error.

NAME

rename – change the name of a file

SYNOPSIS

rename(from, to)
char *from, *to;

DESCRIPTION

Rename causes the link named **from** to be renamed as **to**. If **to** exists, then it is first removed. Both **from** and **to** must be of the same type (that is, both directories or both non-directories), and must reside on the same file system.

Rename guarantees that an instance of **to** always exists, even if the system should crash in the middle of the operation.

WARNING

The system can deadlock if a loop in the file system graph is present. This loop takes the form of an entry in directory `a`, say `a/foo`, being a hard link to directory `b`, and an entry in directory `b`, say `b/bar`, being a hard link to directory `a`. When such a loop exists and two separate processes attempt to perform `rename a/foo b/bar` and `rename b/bar a/foo`, respectively, the system may deadlock attempting to lock both directories for modification. Hard links to directories should be replaced by symbolic links by the system administrator.

DIAGNOSTICS

A 0 value is returned if the operation succeeds, otherwise *rename* returns –1 and the global variable *errno* indicates the reason for the failure.

Rename fails and neither of the argument files is affected if any of the following are true:

[EACCES]	A component of either path prefix denies search permission.
[EACCES]	The requested link requires writing in a directory with a mode that denies write permission.
[EACCESS]	The caller is not the superuser and the parent directory has the sticky bit set and the *to* or the *from* file is not writable by the user and the user does not own the parent directory and the user does not own the *to* or the *from* file.
[EBUSY]	Either the directory named by *to* or the directory named by *from* is the object of a concurrent mount operation.
[EFAULT]	*From* or *to* points outside the process's allocated address space.
[EINVAL]	*From* is a parent directory of *to*.
[EINVAL]	The directory named by *to* is the caller's current working directory.
[EEXIST]	The link named by *to* is a directory containing entries other than dot and dot-dot.
[EIO]	An I/O error occurred while reading from or writing to the file system.

[EISDIR]	An attempt was mad to rename a non-directory file to an existing directory.
[ELOOP]	Too many symbolic links were encountered in translating the *from* pathname or the *to* pathname.
[ENAMETOOLONG]	The length of the *from* pathname or the *to* pathname exceeds PATH_MAX characters.
[ENAMETOOLONG]	The *from* pathname or the *to* pathname referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of the *from* or *to* pathnames was longer than NAME_MAX characters.
[ENOENT]	A component of either path prefix does not exist.
[ENOENT]	The file named by *from* does not exist.
[ENOSPC]	The directory in which the entry for the new name is being placed cannot be extended because there is no space left on the file system containing the directory.
[ENOTDIR]	A component of either path prefix is not a directory.
[ENOTDIR]	An attempt was made to rename a directory to an existing non-directory file.
[EROFS]	The requested link requires writing in a directory on a read-only file system.
[ETXTBSY]	The file named by *to* or the file named by *from* is a pure procedure (shared text) file that is currently being executed.

SEE ALSO

open(2)

NAME

rmdir – remove a directory

SYNOPSIS

int rmdir (path)
char *path;

DESCRIPTION

rmdir removes the directory named by the path name pointed to by **path.** The directory must not have any entries other than "." and "..".

The named directory is removed unless one or more of the following are true:

[EACCES]	Search permission is denied for a component of the path prefix.
[EACCES]	Write permission is denied on the directory containing the directory to be removed.
[EACCES]	The parent directory has the sticky bit set and the parent directory is not owned by the user and the directory is not owned by the user and the directory is not writable by the user and the user is not superuser
[EBUSY]	The directory to be removed is the mount point for a mounted file system.
[EEXIST]	The directory contains entries other than those for "." and "..".
[EFAULT]	**path** points outside the process's allocated address space.
[EINVAL]	The current directory may not be removed.
[EINVAL]	The "." entry of a directory may not be removed.
[EIO]	An I/O error occurred while accessing the file system.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**Path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**Path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	The named directory does not exist.
[ENOTDIR]	A component of the path prefix is not a directory.
[EROFS]	The directory entry to be removed is part of a read-only file system.

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

SEE ALSO

rmdir(1), rm(1), and mkdir(1), mkdir(2), unlink(2)

NAME

semctl – semaphore control operations

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

int semctl (semid, semnum, cmd, arg)
int semid, cmd;
int semnum;
union semun {
int val;
struct semid_ds *buf;
ushort *array;
} arg;
```

DESCRIPTION

semctl provides a variety of semaphore control operations as specified by **cmd**.

The following **cmds** are executed with respect to the semaphore specified by **semid** and **semnum:**

GETVAL	Return the value of semval (see **intro**(2)). {READ}
SETVAL	Set the value of **semval** to **arg.val**. {ALTER} When this cmd is successfully executed, the semadj value corresponding to the specified semaphore in all processes is cleared.
GETPID	Return the value of sempid. {READ}
GETNCNT	Return the value of semncnt. {READ}
GETZCNT	Return the value of semzcnt. {READ}

The following **cmds** return and set, respectively, every **semval** in the set of semaphores.

GETALL	Place **semvals** into array pointed to by **arg.array**. {READ}
SETALL	Set **semvals** according to the array pointed to by **arg.array**. {ALTER} When this cmd is successfully executed the semadj values corresponding to each specified semaphore in all processes are cleared.

The following **cmds** are also available:

IPC_STAT	Place the current value of each member of the data structure associated with **semid** into the structure pointed to by **arg.buf**. The contents of this structure are defined in **intro**(2). {READ}
IPC_SET	Set the value of the following members of the data structure associated with **semid** to the corresponding value found in the structure pointed to by **arg.buf**:

```
sem_perm.uid
```

```
sem_perm.gid
sem_perm.mode /* only low 9 bits */
```

This command can only be executed by a process that has an effective user ID equal to either that of superuser, or to the value of **sem_perm.cuid** or **sem_perm.uid** in the data structure associated with **semid**.

IPC_RMID — Remove the semaphore identifier specified by **semid** from the system and destroy the set of semaphores and data structure associated with it. This command can only be executed by a process that has an effective user ID equal to either that of superuser, or to the value of **sem_perm.cuid** or **sem_perm.uid** in the data structure associated with **semid**.

semctl fails if any of the following are true:

[EACCES] Operation permission is denied to the calling process (see **intro**(2)).

[EFAULT] **arg.buf** points to an illegal address.

[EINVAL] **semid** is not a valid semaphore identifier.

[EINVAL] **semnum** is less than zero or greater than **sem_nsems**.

[EINVAL] **cmd** is not a valid command.

[EPERM] **cmd** is equal to IPC_RMID or IPC_SET and the effective user ID of the calling process is not equal to that of superuser, or to the value of **sem_perm.cuid** or **sem_perm.uid** in the data structure associated with **semid**.

[ERANGE] **cmd** is SETVAL or SETALL and the value to which **semval** is to be set is greater than the system imposed maximum.

SEE ALSO

intro(2), semget(2), semop(2)

DIAGNOSTICS

Upon successful completion, the value returned depends on **cmd** as follows:

GETVAL — The value of **semval**

GETPID — The value of **sempid**

GETNCNT — The value of **semncnt**

GETZCNT — The value of **semzcnt**

All others — A value of 0

Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

semget – get set of semaphores

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

int semget (key, nsems, semflg)
key_t key;
int nsems, semflg;
```

DESCRIPTION

semget returns the semaphore identifier associated with **key**.

A semaphore identifier and associated data structure and set containing **nsems** semaphores (see **intro**(2)) are created for **key** if any of the following is true:

key is equal to IPC_PRIVATE.

key does not already have a semaphore identifier associated with it, and (**semflg** & IPC_CREAT) is true.

Upon creation, the data structure associated with the new semaphore identifier is initialized as follows:

sem_perm.cuid, **sem_perm.uid**, **sem_perm.cgid**, and **sem_perm.gid** are set equal to the effective user ID and effective group ID, respectively, of the calling process.

The low-order 9 bits of **sem_perm.mode** are set equal to the low-order 9 bits of **semflg**.

sem_nsems is set equal to the value of **nsems**.

sem_otime is set equal to 0 and **sem_ctime** is set equal to the current time.

semget fails if any of the following are true:

[EACCES] A semaphore identifier exists for **key**, but operation permission [see **intro**(2)] as specified by the low-order 9 bits of **semflg** would not be granted.

[EEXIST] A semaphore identifier exists for **key** but ((**semflg** & IPC_CREAT) and (**semflg** & IPC_EXCL)) is true.

[EINVAL] **nsems** is either less than or equal to zero or greater than the system-imposed limit.

[EINVAL] A semaphore identifier exists for **key**, but the number of semaphores in the set associated with it is less than **nsems**, and **nsems** is not equal to zero.

[ENOENT] A semaphore identifier does not exist for **key** and (**semflg** & IPC_CREAT) is false.

[ENOSPC] A semaphore identifier is to be created but the system-imposed limit on the maximum number of allowed semaphore identifiers system wide would be exceeded.

[ENOSPC] A semaphore identifier is to be created but the system-imposed limit on the maximum number of allowed semaphores system wide would be exceeded.

SEE ALSO

intro(2), semctl(2), semop(2)

DIAGNOSTICS

Upon successful completion, a non-negative integer, namely a semaphore identifier, is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

semop – semaphore operations

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

int semop (semid, sops, nsops)
int semid;
struct sembuf **sops;
unsigned nsops;
```

DESCRIPTION

semop is used to automatically perform an array of semaphore operations on the set of semaphores associated with the semaphore identifier specified by **semid**. **sops** is a pointer to the array of semaphore-operation structures. **nsops** is the number of such structures in the array. The contents of each structure includes the following members:

```
short  sem_num;  /* semaphore number */
short  sem_op;   /* semaphore operation */
short  sem_flg;  /* operation flags */
```

Each semaphore operation specified by **sem_op** is performed on the corresponding semaphore specified by **semid** and **sem_num**.

sem_op specifies one of three semaphore operations as follows:

If **sem_op** is a negative integer, one of the following occurs: (ALTER)

If **semval** (see **intro**(2)) is greater than or equal to the absolute value of **sem_op**, the absolute value of **sem_op** is subtracted from **semval**. Also, if (**sem_flg** & SEM_UNDO) is true, the absolute value of **sem_op** is added to the calling process's **semadj** value (see **exit**(2)) for the specified semaphore.

If semval is less than the absolute value of **sem_op** and (**sem_flg** & IPC_NOWAIT) is true, **semop** returns immediately.

If **semval** is less than the absolute value of **sem_op** and (**sem_flg** & IPC_NOWAIT) is false, **semop** increments the semncnt associated with the specified semaphore and suspends execution of the calling process until one of the following conditions occur:

semval becomes greater than or equal to the absolute value of **sem_op**. When this occurs, the value of **semncnt** associated with the specified semaphore is decremented, the absolute value of **sem_op** is subtracted from **semval** and, if (sem_flg & **SEM_UNDO**) is true, the absolute value of **sem_op** is added to the calling process's **semadj** value for the specified semaphore.

The **semid** for which the calling process is awaiting action is removed from the system (see **semctl**(2)). When this occurs, **errno** is set equal to EIDRM, and a value of –1 is

returned.

The calling process receives a signal that is to be caught. When this occurs, the value of **semncnt** associated with the specified semaphore is decremented, and the calling process resumes execution in the manner prescribed in **signal**(2).

If **sem_op** is a positive integer, the value of **sem_op** is added to **semval** and, if (**sem_flg** & SEM_UNDO) is true, the value of **sem_op** is subtracted from the calling process's **semadj** value for the specified semaphore. {ALTER}

If **sem_op** is zero, one of the following occurs: {READ}

If **semval** is zero, **semop** returns immediately.

If **semval** is not equal to zero and (**sem_flg** & PC_NOWAIT) is true, **semop** returns immediately.

If **semval** is not equal to zero and (**sem_flg** & IPC_NOWAIT) is false, **semop** increments the **semzcnt** associated with the specified semaphore and suspends execution of the calling process until one of the following occurs:

semval becomes zero, at which time the value of **semzcnt** associated with the specified semaphore is decremented.

The **semid** for which the calling process is awaiting action is removed from the system. When this occurs, **errno** is set equal to EIDRM, and a value of −1 is returned.

The calling process receives a signal that is to be caught. When this occurs, the value of **semzcnt** associated with the specified semaphore is decremented, and the calling process resumes execution in the manner prescribed in **signal**(2).

semop fails if any of the following are true for any of the semaphore operations specified by **sops**:

[EACCES] Operation permission is denied to the calling process [see **intro**(2)]

[EAGAIN] The operation would result in suspension of the calling process but (**sem_flg** & IPC_NOWAIT) is true.

[EFAULT] **sops** points to an illegal address.

[EFBIG] **sem_num** is less than zero or greater than or equal to the number of semaphores in the set associated with **semid**.

[EIDRM] **semop** returned due to the removal of **semid** from the system.

[EINVAL] **semid** is not a valid semaphore identifier.

[EINVAL] The number of individual semaphores for which the calling process requests a **SEM_UNDO** would exceed the limit.

[EINTR] **semop** returned due to the receipt of a signal.

[E2BIG] **nsops** is greater than the system-imposed maximum.

[ENOSPC] The limit on the number of individual processes requesting an **SEM_UNDO** would be exceeded.

[ERANGE] An operation would cause a **semval** to overflow the system-imposed limit.

[ERANGE] An operation would cause a **semadj** value to overflow the system-imposed limit.

Upon successful completion, the value of **sempid** for each semaphore specified in the array pointed to by **sops** is set equal to the process ID of the calling process.

SEE ALSO

exec(2), exit(2), fork(2), intro(2), semctl(2), semget(2)

DIAGNOSTICS

If **semop** returns due to the receipt of a signal, a value of –1 is returned to the calling process and **errno** is set to EINTR. If it returns due to the removal of a **semid** from the system, a value of –1 is returned and **errno** is set to EIDRM.

Upon successful completion, a value of zero is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

setpgid – manipulate signal sets

SYNOPSIS

#include <sys/types.h>

int setpgid (pid, pgid)
pid_t pid, pgid;

DESCRIPTION

The **setpgid function** is used to either join an existing process group or create a new process group within the session of the calling process. The process group ID of a session leader does not change. Upon successful completion, the process group ID of the process with a process ID that matches **pid** shall be set to **pgid**. As a special case, if **pid** is zero, the process ID of the calling process is used. Also, if **pgid** is zero, the process ID of the indicated process is used.

On successful completion, **setpgid** returns a value of zero. Otherwise, a value of -1 is returned and **errno** is set to indicate the error.

If any of the following conditions occur, **setpgid** returns -1 and sets **errno** to the corresponding value:

[EACCES] The value of **pid** matches the process ID of a child process of the calling process and the child process has successfully executed one of the functions.

[EINVAL] The value of **pgid** is less than zero or is greater than MAXPID.

[ENOSYS] The **setpgid** function is not supported.

[EPERM] The process indicated by **pid** is a session leader.
The value of **pid** is valid but matches the process ID of a child of the calling process and the child process is not in the same session as the calling process.
The value of **pgid** does not match the process ID of the indicated by **pid**, and there is no process with a process group with a process group ID that matches the value of the **pgid** argument in the same session as the calling process.

[ESRCH] The value of the **pid** argument does not match the process ID of the calling process or of a child process of the calling process.

SEE ALSO

getpgrp(2), setsid(2), tcsetpgrp(2), exec(2)

NAME

setpgrp – set process group ID

SYNOPSIS

int setpgrp ()

DESCRIPTION

setpgrp sets the process group ID of the calling process to the process ID of the calling process and returns the new process group ID.

SEE ALSO

exec(2), fork(2), getpid(2), intro(2), kill(2), signal(2), setsid(2)

DIAGNOSTICS

setpgrp returns the value of the new process group ID.

NAME

setsid – create session and set process group ID

SYNOPSIS

#include <sys/types.h>

pid_t setsid()

DESCRIPTION

If the calling process is not a process group leader, **setsid** creates a new session. The calling process is the session leader of this new session, the process group leader of a new process group, and has no controlling terminal. The process group ID of the calling process is set equal to the process ID of the calling process. The calling process is the only process in the new process group and the only process in the new session.

Upon sucessful completion, **setsid** returns the value of the process group ID of the calling process.

If the following condition occurs, **setsid** returns -1 and sets **errno** to the corresponding value:

[EPERM] The calling process is already a process group leader, or the process group ID of a process other than the calling process matches the process ID of the calling process.

SEE ALSO

exec (2), exit(2), fork(2), getpid(2), kill(1), setuid(2), sigaction(2)

NAME

setuid, setgid – set user and group IDs

SYNOPSIS

int setuid (uid)
int uid;

int setgid (gid)
int gid;

DESCRIPTION

setuid (**setgid**) is used to set the real user (group) ID and effective user (group) ID of the calling process.

If the effective user ID of the calling process is superuser, the real user (group) ID, effective user (group) ID, and saved set-user (group) ID are set to **uid** (**gid**).

If the effective user ID of the calling process is not superuser, but its real user (group) ID is equal to **uid** (**gid**), the effective user (group) ID is set to **uid** (**gid**).

If the effective user ID of the calling process is not superuser, but the saved set-user (group) ID from **exec**(2) is equal to **uid** (**gid**), the effective user (group) ID is set to **uid** (**gid**).

setuid (**setgid**) fails if the following are true:

[EINVAL] The **uid** is out of range.

[EPERM] Real user (group) ID of the calling process is not equal to **uid** (**gid**) and its effective user ID is not superuser.

SEE ALSO

getuid(2), intro(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

shmctl – shared memory control operations

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int shmctl (shmid, cmd, buf)
int shmid, cmd;
struct shmid_ds *buf;
```

DESCRIPTION

shmctl provides a variety of shared memory control operations as specified by **cmd**. The following **cmds** are available:

IPC_STAT — Place the current value of each member of the data structure associated with **shmid** into the structure pointed to by **buf**. The contents of this structure are defined in **intro**(2). {READ}

IPC_SET — Set the value of the following members of the data structure associated with **shmid** to the corresponding value found in the structure pointed to by **buf**:

```
shm_perm.uid
shm_perm.gid
shm_perm.mode /* only low 9 bits */
```

This **cmd** can only be executed by a process that has an effective user ID equal to that of super user, or to the value of **shm_perm.cuid** or **shm_perm.uid** in the data structure associated with **shmid**.

IPC_RMID — Remove the shared memory identifier specified by **shmid** from the system and destroy the shared memory segment and data structure associated with it. If the shared memory segment is still attached to a process, the shared memory identifier will no longer be available for attachment by other processes and will be removed from the system only when the associated process terminates or detaches the segment. This **cmd** can only be executed by a process that has an effective user ID equal to that of super user, or to the value of **shm_perm.cuid** or **shm_perm.uid** in the data structure associated with **shmid**.

SHM_LOCK — Lock the shared memory segment specified by **shmid** in memory. This **cmd** is not supported.

SHM_UNLOCK — Unlock the shared memory segment specified by **shmid**. This **cmd** is not supported.

shmctl fails if one or more of the following are true:

[EACCES] — **cmd** is equal to IPC_STAT and {READ} operation permission is denied to the calling process (see **intro**(2)).

[EFAULT] — **buf** points to an illegal address.

[EINVAL]	**shmid** is not a valid shared memory identifier.
[EINVAL]	**cmd** is not a valid command.
[EPERM]	**cmd** is equal to IPC_RMID or IPC_SET and the effective user ID of the calling process is not equal to that of super user, or to the value of **shm_perm.cuid** or **shm_perm.uid** in the data structure associated with **shmid**.
[EPERM]	**cmd** is equal to SHM_LOCK or SHM_UNLOCK and the effective user ID of the calling process is not equal to that of super user.

SEE ALSO

shmget(2), shmop(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error.

NOTES

The user must explicitly remove shared memory segments after the last reference to them has been removed.

NAME

shmget – get shared memory segment identifier

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int shmget (key, size, shmflg)
key_t key;
int size, shmflg;
```

DESCRIPTION

shmget returns the shared memory identifier associated with **key**.

A shared memory identifier and associated data structure and shared memory segment of at least **size** bytes (see **intro**(2)) are created for **key** if one of the following are true:

key is equal to IPC_PRIVATE.

key does not already have a shared memory identifier associated with it, and (**shmflg** & IPC_CREAT) is true.

Upon creation, the data structure associated with the new shared memory identifier is initialized as follows:

Shm_perm.cuid, **shm_perm.uid**, **shm_perm.cgid**, and **shm_perm.gid** are set equal to the effective user ID and effective group ID, respectively, of the calling process.

The low-order 9 bits of **shm_perm.mode** are set equal to the low-order 9 bits of **shmflg**. **Shm_segsz** is set equal to the value of **size**.

Shm_lpid, **shm_nattch**, **shm_atime**, and **shm_dtime** are set equal to 0.

Shm_ctime is set equal to the current time.

shmget fails if any of the following are true:

[EACCES] A shared memory identifier exists for **key** but operation permission (see **intro**(2)) as specified by the low-order 9 bits of **shmflg** would not be granted.

[EEXIST] A shared memory identifier exists for **key** but ((**shmflg** & IPC_CREAT) and (**shmflg** & IPC_EXCL)) is true.

[EINVAL] **size** is less than the system-imposed minimum or greater than the system-imposed maximum.

[EINVAL] A shared memory identifier exists for **key** but the size of the segment associated with it is less than **size** and **size** is not equal to zero.

[ENOENT] A shared memory identifier does not exist for **key** and (**shmflg** & IPC_CREAT) is false.

[ENOMEM] A shared memory identifier and associated shared memory segment are to be created but the amount of available memory is not sufficient to fill the request.

[ENOSPC] A shared memory identifier is to be created but the system-imposed limit on the maximum number of allowed shared memory identifiers system wide would be exceeded.

SEE ALSO

intro(2), shmctl(2), shmop(2)

DIAGNOSTICS

Upon successful completion, a non-negative integer, namely a shared memory identifier is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error.

NOTES

The user must explicitly remove shared memory segments after the last reference to them has been removed.

NAME

shmop, shmat, shmdt – shared memory operations

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

char *shmat (shmid, shmaddr, shmflg)
int shmid;
char *shmaddr;
int shmflg;

int shmdt (shmaddr)
char *shmaddr;
```

DESCRIPTION

shmat attaches the shared memory segment associated with the shared memory identifier specified by **shmid** to the data segment of the calling process. The segment is attached at the address specified by one of the following criteria:

If **shmaddr** is equal to zero, the segment is attached at the first available address as selected by the system.

If **shmaddr** is not equal to zero and (**shmflg** & SHM_RND) is true, the segment is attached at the address given by (**shmaddr** - (**shmaddr** modulus SHMLBA)).

If **shmaddr** is not equal to zero and (**shmflg** & SHM_RND) is false, the segment is attached at the address given by **shmaddr**.

shmdt detaches from the calling process's data segment the shared memory segment located at the address specified by **shmaddr**.

The segment is attached for reading if (**shmflg** & SHM_RDONLY) is true {READ}, otherwise it is attached for reading and writing {READ/WRITE}.

shmat fails and does not attach the shared memory segment if one or more of the following are true:

[EACCES]	Operation permission is denied to the calling process [see **intro**(2)].
[EINVAL]	**shmid** is not a valid shared memory identifier.
[EINVAL]	**shmaddr** is not equal to zero, and the value of (**shmaddr** - (**shmaddr** modulus SHMLBA)) is an illegal address.
[EINVAL]	**shmaddr** is not equal to zero, (**shmflg** & SHM_RND) is false, and the value of **shmaddr** is an illegal address.
[EINVAL]	**shmdt** fails and does not detach the shared memory segment if **shmaddr** is not the data segment start address of a shared memory segment.
[EMFILE]	The number of shared memory segments attached to the calling process would exceed the system-imposed limit.
[ENOMEM]	The available data space is not large enough to accommodate the shared memory segment.

SEE ALSO

exec(2), exit(2), fork(2), intro(2), shmctl(2), shmget(2)

DIAGNOSTICS

Upon successful completion, the return values are as follows:

shmat returns the data segment start address of the attached shared memory segment.

shmdt returns a value of 0.

Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NOTES

The user must explicitly remove shared memory segments after the last reference to them has been removed.

NAME

sigaction – examine and change signal actions

SYNOPSIS

```
#include <signal.h>

int sigaction (sig, act, oact)
int sig;
struct sigaction *act, *oact;
```

DESCRIPTION

The **sigaction** function allows the calling process to examine or specify (or both) the action to be associated with a specific signal. The argument **sig** specifies the signal; acceptable values are defined in *<signal.h>*.

The structure **sigaction**, used to describe an action to be taken, is defined in *<signal.h>* to include following members:

Member Type	Member Name	Description
void (*)()	sa_handler	SIG_DFL, SIG_IGN, or pointer to a function
sigset_t	sa_mask	Additional set of signals to be blocked during execution of signal-catching function.
int	sa_flags	Special flags to affect behavior of signal

If **act** is not NULL, it points to a structure specifying the action to be associated with the specified signal. If **oact** is not NULL, the action previously associated with the signal is stored in the location pointed to by the argument **oact**. If **act** is NULL, signal handling is unchanged by this function call; thus, the call can be used to inquire about the current handling of a given signal. The **sa_handler** field of the **sigaction** structure identifies the action to be associated with the specified signal. If the **sa_handler** field specifies a signal-catching function, the **sa_mask** field identifies a set of signals that are added to the process's signal mask before the signal-catching function is invoked. The SIGKILL and SIGSTOP signals are not added to the signal mask using this mechanism; this restriction is enforced by the system without causing an error to be indicated.

The **sa_flags** field can be used to modify the behavior of the specified signal.

The flag bit SA_NOCLDSTOP (symbolic constant), defined in *<signal.h>*, can be set in **sa_flags** to prevent SIGCHLD from being generated when children stop.

If **sig** is SIGCHLD and the SA_NOCLDSTOP flag is not set in **sa_flags**, a SIGCHLD signal is generated for the calling process whenever any of its child processes stop. If **sig** is SIGCHLD and the SA_NOCLDSTOP flag is set in **sa_flags**, a SIGCHLD signal is not generated in this way.

When a signal is caught by a signal-catching function installed by **sigaction**, a new signal mask is calculated and installed for the duration of the signal-catching function (or until a call to either **sigprocmask** or **sigsuspend**). This mask is formed by taking the union of the current signal mask and the value of the **sa_mask** for the signal being delivered, and then including the signal being delivered. If and when the user's signal handler returns

normally, the original signal mask is restored.

Once an action is installed for a specific signal, it remains installed until another action is explicitly requested (by another call to **sigaction**), or until one of the **exec** functions is called.

If the previous action for sig had been established by the **signal**(2) function, the values of the fields returned in the structure pointed to by **oact** are unspecified, and in particular **oact->sv_handler** is not necessarily the same value passed to **signal** function. However, if a pointer to the same structure or a copy thereof is passed to a subsequent call to **sigaction** via the act argument, handling of the signal is as if the original call to **signal** were repeated.

If **sigaction** function fails, no new signal handler is installed.

On successful completion, a value of zero is returned. Otherwise, a value of -1 is returned and **errno** is set to indicate the error.

If the following condition occurs, **sigaction** returns -1 and sets **errno** to the corresponding value:

[EINVAL] The value of the **sig** argument is an invalid or unsupported signal number, or an attempt was made to catch a signal that cannot be caught or to ignore a signal that cannot be ignored. Refer to *<signal.h>*.

[EFAULT] **act** or **oact** points outside the allocated address space.

SEE ALSO

kill(1), sigprocmask(2), sigsuspend(2)

NAME

sigemptyset, sigfillset, sigaddset, sigdelset, sigismember – manipulate signal sets

SYNOPSIS

```
#include <signal.h>

int sigemptyset (set)
sigset_t *set;

int sigfillset (set)
sigset_t *set;

int sigaddset (set,signo)
sigset_t *set;
int signo;

int sigdelset (set, signo)
sigset_t *set;
int signo;

int sigismember (set, signo)
sigset_t *set;
int signo;
```

DESCRIPTION

The **sigsetops** primitives manipulate sets of signals. They operate on data objects addressable by the application, not on any set of signals known to the system, such as the set blocked from delivery to a process or the set pending for a process.

The **sigemptyset** function initializes the signal set pointed to by **set**, such that all signals defined in this standard are excluded.

The **sigfillset** function initializes the signal set pointed to by **set**, such that all signals defines in this standard are included.

Applications call either **sigemptyset** or **sigfillset** at least once for each object of type **sigset_t** prior to any other object. If such an object is not initialized in this way, but is supplied as an argument to any of the **sigaddset, sigdelset, sigismember, sigaction, sigprocmask, sigpending**, or **sigsuspend** functions, the results are undefined.

The **sigaddset** and **sigdelset** functions respectively add and delete the individual signal specified by the value of **signo** from the signal set pointed to by the argument set.

The **sigismember** function tests whether the signal specified by the value of **signo** is a member of the set pointed to by the argument set.

On successful completion, **sigismember** function returns a value of one if the specified signal is a member of the specified set, or a value of zero if it is not. On successful completion, the other functions return a value of zero. If an error is detected for any of the above functions, a value of -1 is returned and **errno** is set to indicate the error.

If the following condition is detected, **sigaddset, sigelset**, and **sigismember** return -1 and set **errno** to the corresponding value:

[EINVAL] The value of **signo** is an invalid or unsupported signal number.

SEE ALSO

sigaction(2), sigpending(2), sigprocmask(2), sigsupend(2)

NAME

signal – specify what to do upon receipt of a signal

SYNOPSIS

```
#include <signal.h>

void (*signal (sig, func))()
int sig;
void (*func)();
```

DESCRIPTION

signal allows the calling process to choose one of three ways in which it is possible to handle the receipt of a specific signal. **sig** specifies the signal and **func** specifies the choice.

sig can be assigned any one of the following except SIGKILL (the number in brackets corresponds to the number of the note below–see **NOTES**):

SIGHUP	01	hangup
SIGINT	02	interrupt
SIGQUIT	03[1]	quit
SIGILL	04[1]	illegal instruction (not reset when caught)
SIGTRAP	05[1]	trace trap (not reset when caught)
SIGIOT	06[1]	IOT instruction
SIGABRT	06[1]	abort and dump core
SIGEMT	07[1]	EMT instruction
SIGFPE	08[1]	floatingpoint exception
SIGKILL	09	kill (cannot be caught or ignored)
SIGBUS	10[1]	bus error
SIGSEGV	11[1]	segmentation violation
SIGSYS	12[1]	bad argument to system call
SIGPIPE	13	write on a pipe with no one to read it
SIGALRM	14	alarm clock
SIGTERM	15	software termination signal
SIGUSR1	16	user-defined signal 1
SIGUSR2	17	user-defined signal 2
SIGCLD	18[2]	death of a child
SIGPWR	19[2]	power fail
SIGWINCH	20[2]	window change
SIGPOLL	22[3]	selectable event pending
SIGSTOP	23[4]	stop process(cannot be caught or ignored)
SIGTSTP	24[4]	interactive stop signal
SIGCONT	25[5]	continue process
SIGTTIN	26[4]	input from background
SIGTTOU	27[4]	output from background

func is assigned one of three values: SIG_DFL, SIG_IGN, or a **function address**. SIG_DFL and SIG_IGN are defined in the include file *<signal.h>*. Each is a macro that expands to a constant expression of type pointer to function returning **void**, and has a unique value that matches no declarable function.

The actions prescribed by the values of **func** are as follows:

SIG_DFL – terminate process upon receipt of a signal

Upon receipt of the signal **sig**, the receiving process is to be terminated with all of the consequences outlined in **exit**(2). See NOTE [1] below.

SIG_IGN – ignore signal

The signal **sig** is to be ignored. Note that the signal SIGKILL cannot be ignored.

function address – catch signal

Upon receipt of the signal **sig**, the receiving process is to execute the signal-catching function pointed to by **func**. The signal number **sig** is passed as the only argument to the signal-catching function. Additional arguments are passed to the signal-catching function for hardware-generated signals. Before entering the signal-catching function, the value of **func** for the caught signal is set to **SIG_DFL** unless the signal is SIGILL, SIGTRAP, or SIGPWR.

Upon return from the signal-catching function, the receiving process resumes execution at the point it was interrupted.

When a signal that is to be caught occurs during a **read**(2), a **write**(2), an **open**(2), or an **ioctl**(2) system call on a slow device (like a terminal; but not a file), during a **pause**(2) system call, or during a **wait**(2) system call that does not return immediately due to the existence of a previously stopped or zombie process, the signal catching function will be executed and then the interrupted system call may return a –1 to the calling process with **errno** set to EINTR.

signal will not catch an invalid function argument, **func**, and results are undefined when an attempt is made to execute the function at the bad address.

Note that the signal SIGKILL cannot be caught.

A call to **signal** cancels a pending signal **sig** except for a pending SIGKILL signal.

signal fails if **sig** is an illegal signal number, including SIGKILL. [EINVAL]

NOTES

[1] If SIG_DFL is assigned for these signals, in addition to the process being terminated, a core image is constructed in the current working directory of the process, if the following conditions are met:

The effective user ID and the real user ID of the receiving process are equal.

An ordinary file named *core* exists and is writable or can be created. If the file must be created, it will have the following properties:

a mode of 0666 modified by the file creation mask [see *umask*(2)]

a file owner ID that is the same as the effective user ID of the receiving process.

a file group ID that is the same as the group ID of the directory in which the file is created.

[2] For the signals SIGCLD and SIGPWR, **func** is assigned one of three values: SIG_DFL, SIG_IGN, or a **function address**. The actions prescribed by these values are:

SIG_DFL - ignore signal
The signal is to be ignored.

SIG_IGN - ignore signal
The signal is to be ignored. Also, if **sig** is SIGCLD, the calling process's child processes will not create zombie processes when they terminate (see **exit**(2)).

function address - catch signal
If the signal is SIGPWR, the action to be taken is the same as that described above for **func** equal to **function address**. The same is true if the signal is SIGCLD with one exception: while the process is executing the signal-catching function, any received SIGCLD signals are ignored. (This is the default action.)

In addition, SIGCLD affects the **wait**, and **exit** system calls as follows:

wait If the **func** value of SIGCLD is set to SIG_IGN and a **wait** is executed, the **wait** blocks until all of the calling process's child processes terminate; it then returns a value of –1 with **errno** set to ECHILD.

exit If in the exiting process's parent process the **func** value of SIGCLD is set to SIG_IGN, the exiting process will not create a zombie process.

When processing a pipeline, the shell makes the last process in the pipeline the parent of the proceeding processes. A process that may be piped into in this manner (and thus become the parent of other processes) should take care not to set SIGCLD to be caught.

[3] SIGPOLL is issued when a file descriptor corresponding to a STREAMS [see **intro**(2)] file has a selectable event pending. A process must specifically request that this signal be sent using the I_SETSIG **ioctl** call. Otherwise, the process will never receive SIGPOLL.

[4] The default action for these signal is to stop the process. When any of these stop signals are generated for a process, a pending SIGCONT is discarded.

[5] The action for SIGCONT is to continue the process if it is currently stopped, otherwise the signal is ignored, even if SIGCONT is being blocked or ignored. Pending stop signals are discarded.

SEE ALSO

kill(1), intro(2), kill(2), pause(2), ptrace(2), wait(2), sigset(2), setjmp(3C), sigcontext(2seq)

DIAGNOSTICS

Upon successful completion, **signal** returns the previous value of **func** for the specified signal **sig**. Otherwise, a value of SIG_ERR is returned and **errno** is

set to indicate the error. SIG_ERR is defined in the include file <*signal.h*>.

NAME

sigpending – examine pending signals

SYNOPSIS

```
#include <signal.h>

int sigpending (set)
sigset_t *set;
```

DESCRIPTION

The **sigpending** function stores the set of signals that are blocked from delivery and pending for the calling process, in the space pointed to by **set.**

sigpending will fail if the following is true:

[EFAULT] **set** points to an illegal address.

SEE ALSO

sigprocmask(2), sigemptyset(2), sigaddset(2), sigdelset(2)

DIAGNOSTICS

On successful completion, a value of zero is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

sigprocmask – examine and change blocked signals

SYNOPSIS

```
#include <signal.h>

int sigprocmask (how, set, oset)
int how;
sigset_t *set, *oset;
```

DESCRIPTION

The **sigprocmask** function is used to examine or change (or both) the calling process's signal mask. If the value of **set** is not NULL, it points to a set of signals to be used to change the currently blocked set.

The value of **how** indicates the manner in which the set is changed, and consists of one of the following values, as defined in the header *<signal.h>*:

SIG_BLOCK	The resulting set shall be the union of the current set and the signal set pointed to by **set**.
SIG_UNBLOCK	The resulting set shall be the intersection of the current set and the complement of the signal set pointed to by **set**.
SIG_SETMASK	The resulting set shall be the signal set pointed to by the argument set.

If **oset** is not NULL, the previous mask is stored in the space pointed to by **oset**. If the value of **set** is NULL, the value of **how** is not significant and the process's signal mask is unchanged by this function call; thus, the call can be used to inquire about currently blocked signals.

If there are any pending unblocked signals after the call to **sigprocmask**, at least one of those signals is delivered before **sigprocmask** returns.

It is possible to block the SIGKILL and SIGSTOP signals; this is enforced by the system without causing an error to be indicated.

If any of the SIGFPE, SIGLL or SIGSEGV signals are generated while they are blocked, the result is undefined, unless the signal was generated by a call to the **kill** function.

If **sigprocmask** fails, the process's signal mask is not changed by this function call.

On sucessful completion, a value of zero is returned. Otherwise, a value of -1 is returned and **errno** is set to indicate the error.

If the following condition occurs, **sigprocmask** returns -1 and sets **errno** to the corresponding value:

[EINVAL] The value of **how** is not equal to one of the defined values.

SEE ALSO

sigaction(2), sigpending(2), sigsuspend(2)

NAME

sigset, sighold, sigrelse, sigignore, sigpause – signal management

SYNOPSIS

```
#include <signal.h>

void (*sigset (sig, func))( )
int sig;
void (*func)( );

int sighold (sig)
int sig;

int sigrelse (sig)
int sig;

int sigignore (sig)
int sig;

int sigpause (sig)
int sig;
```

DESCRIPTION

These functions provide signal management for application processes. **sigset** specifies the system signal action to be taken upon receipt of signal **sig**. This action is either calling a process signal-catching handler **func** or performing a system-defined action.

sig can be assigned any one of the following values except SIGKILL. Each value of **sig** is a macro, defined in *<signal.h>*, that expands to an integer constant expression.

SIGHUP	hangup
SIGINT	interrupt
SIGQUIT*	quit
SIGILL*	illegal instruction
SIGTRAP*	trace trap
SIGABRT*	abort
SIGEMT*	emulator trap (not set by this system)
SIGFPE*	floatingpoint exception
SIGKILL	kill (can not be caught or ignored)
SIGBUS*	Bus fault (set by this system for an NMI in I/O space)
SIGSEGV*	memory protection fault.
SIGSYS*	bad argument to system call
SIGPIPE	write on a pipe with no one to read it
SIGALRM	alarm clock
SIGTERM	software termination signal
SIGUSR1	user-defined signal 1
SIGUSR2	user-defined signal 2
SIGCLD	death of a child (see **WARNING** below)

SIGPWR	power fail (see **WARNING** below)
SIGWINCH	window size change (see **termio(7)**)
SIGPOLL	selectable event pending (see **NOTES** below)
SIGSTOP	stop a process (not set by the system)
SIGTSTP	the tty sub-system requests the process to stop.
SIGCONT	continue a stopped process.
SIGTTIN	the tty sub-system requests stopping on input.
SIGTTOU	the tty sub-system requests stopping on output.

See below under SIG_DFL regarding asterisks (*) in the above list.

The following values for the system-defined actions of **func** are also defined in *<signal.h>*. Each is a macro that expands to a constant expression of type pointer to function returning **void** and has a unique value that matches no declarable function.

SIG_DFL – default system action

Upon receipt of the signal **sig**, the receiving process is to be terminated with all of the consequences outlined in **exit**(2). In addition a core image is made in the current working directory of the receiving process if **sig** is one for which an asterisk appears in the above list *and* the following conditions are met:

The effective user ID and the real user ID of the receiving process are equal.

An ordinary file named *core* exists and is writable or can be created. If the file must be created, it will have the following properties:

a mode of 0666 modified by the file creation mask (see *umask*(2))

a file owner ID that is the same as the effective user ID of the receiving process.

a file group ID that is the same as the group ID of the directory in which the file is created.

SIG_IGN – ignore signal

Any pending signal **sig** is discarded and the system signal action is set to ignore future occurrences of this signal type.

SIG_HOLD – hold signal

The signal **sig** is to be held upon receipt. Any pending signal of this type remains held. Only one signal of each type is held.

Otherwise, **func** must be a pointer to a function, the signal-catching handler, that is to be called when signal **sig** occurs. In this case, **sigset** specifies that the process will call this function upon receipt of signal **sig**. Any pending signal of this type is released. This handler address is retained across calls to the other signal management functions listed here.

When a signal occurs, the signal number **sig** is passed as the only argument to the signal-catching handler. Before calling the signal-catching handler, the system signal action is set to SIG_HOLD. During normal return from the signal-catching handler, the system signal action is restored to **func** and any held signal of this type released. If a non-local **goto** (**longjmp**) is taken, then

sigrelse must be called to restore the system signal action and release any held signal of this type.

In general, upon return from the signal-catching handler, the receiving process will resume execution at the point it was interrupted. However, when a signal is caught during a **read**(2), a **write**(2), an **open**(2), or an **ioctl**(2) system call during a **sigpause** system call, or during a **wait**(2) system call that does not return immediately due to the existence of a previously stopped or zombie process, the signal-catching handler is executed and then the interrupted system call may return a –1 to the calling process with **errno** set to EINTR.

sighold and **sigrelse** are used to establish critical regions of code. **sighold** is analogous to raising the priority level and deferring or holding a signal until the priority is lowered by **sigrelse**. **sigrelse** restores the system signal action to that specified previously by **sigset**.

sigignore sets the action for signal **sig** to SIG_IGN (see above).

sigpause suspends the calling process until it receives a signal, the same as **pause**(2). However, if the signal **sig** had been received and held, it is released and the system signal action taken. This system call is useful for testing variables that are changed on the occurrence of a signal. The correct usage is to use **sighold** to block the signal first, then test the variables. If they have not changed, then call **sigpause** to wait for the signal. *sigset* fails if one or more of the following are true:

[EINVAL] **sig** is an illegal signal number (including **SIGKILL**) or the default handling of **sig** cannot be changed.

[EINTR] A signal was caught during the system call

DIAGNOSTICS

Upon successful completion, **sigset** returns the previous value of the system signal action for the specified signal **sig**. Otherwise, a value of **SIG_ERR** is returned and **errno** is set to indicate the error. **SIG_ERR** is defined in *<signal.h>*.

For the other functions, upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

SEE ALSO

kill(2), pause(2), signal(2), wait(2), setjmp(3C)

WARNINGS

Three signals that behave differently from the signals described above exist in this release of the operating system:

SIGCLD	death of a child
SIGPWR	power fail
SIGCONT	continue process

For these signals, **func** is assigned one of three values: SIG_DFL, SIG_IGN, or a **function address**. The actions prescribed by these values are as follows:

SIG_DFL - ignore signal
The signal is to be ignored.

SIG_IGN - ignore signal
The signal is to be ignored. Also, if **sig** is SIGCLD, the calling process's child processes will not create zombie processes when they terminate (see **exit**(2)).

function address - catch signal
If the signal is SIGPWR, the action to be taken is the same as that described above for **func** equal to **function address**. The same is true if the signal is SIGCLD with one exception: while the process is executing the signal-catching function, any received SIGCLD signals is ignored. (This is the default action.)

SIGCLD affects two other system calls **wait**(2), and **exit**(2), in the following ways:

wait If the **func** value of SIGCLD is set to SIG_IGN and a **wait** is executed, the **wait** blocks until all of the calling process's child processes terminate; and then returns a value of –1 with **errno** set to ECHILD.

exit If in the exiting process's parent process the **func** value of SIGCLD is set to SIG_IGN , the exiting process will not create a zombie process.

When processing a pipeline, the shell makes the last process in the pipeline the parent of the proceeding processes. A process that may be piped into in this manner (and thus become the parent of other processes) should take care not to set SIGCLD to be caught.

NOTES

SIGPOLL is issued when a file descriptor corresponding to a STREAMS (see **intro**(2)) file has a selectable event pending. A process must specifically request that this signal be sent using the I_SETSIG **ioctl**(2) call (see **streamio**(7)). Otherwise, the process never receives SIGPOLL.

For portability, applications should use only the symbolic names of signals rather than their values and use only the set of signals defined here. The action for the signal SIGKILL can not be changed from the default system action.

Also, additional information is available to the signal-catching handler for hardware-generated signals by using **sigcontext**(2seq). For certain hardware-generated signals, it may not be possible to resume execution at the point of interruption.

The signal type SIGSEGV is reserved for the condition that occurs on an invalid access to a data object.

The other signal management functions, **signal**(2) and **pause**(2), should not be used in conjunction with these routines for a particular signal type.

NAME

sigsuspend – wait for a signal

SYNOPSIS

```
#include <signal.h>

int sigsuspend (sigmask)
sigset_t *sigmask;
```

DESCRIPTION

The **sigsuspend** function replaces the process's signal mask with the set of signals pointed to by **sigmask**, and then suspends the process until delivery of a signal whose action is either to execute a signal-catching function or to terminate the process.

If the action is to terminate the process, **sigsuspend** does not return. If the action is to execute a signal-catching function, **sigsuspend** returns after the signal-catching function returns, with the signal mask restored to the set that existed prior to the **sigsuspend** call.

It is not possible to block signals that cannot be ignored, as documented in *<signal.h>*; this is enforced by the system without causing an error to be indicated.

Since **sigsuspend** suspends process execution indefinitely, there is no successful completion return value. A value of -1 is returned and **errno** is set to indicate the error.

If the following condition occurs, **sigsuspend** returns -1 and sets **errno** to the corresponding value:

[EINTR] A signal is caught process and control is returned from the signal-catching function.

SEE ALSO

pause(2), sigaction(2), sigpending(2), sigprocmask(2)

NAME

stat, lstat, fstat – get file status

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>

int stat (path, buf)
char *path;
struct stat *buf;

int lstat (path, buf)
char *path;
struct stat *buf;

int fstat (fildes, buf)
int fildes;
struct stat *buf;
```

DESCRIPTION

path points to a pathname naming a file. Read, write, or execute permission of the named file is not required, but all directories listed in the pathname leading to the file must be searchable. **stat** obtains information about the named file.

lstat is like **stat** except in the case where the named file is a symbolic link, in which case **lstat** returns information about the link, while **stat** returns information about the file the link references.

buf is a pointer to a **stat** structure into which information is placed concerning the file or symbolic link.

fstat obtains information about an open file known by the file descriptor **fildes**, obtained from a successful **open**, **creat**, **dup**, **fcntl**, or **pipe** system call.

buf is a pointer to a **stat** structure into which information is placed concerning the file.

The contents of the structure pointed to by **buf** include the following members:

```
ushort st_mode;     /* File mode [see mknod (2)] */
ino_t st_ino;       /* Inode number */
dev_t st_dev;       /* ID of device containing */
                    /* a directory entry for this file */
dev_t st_rdev;      /* ID of device */
                    /* This entry is defined only for */
                    /* character special or block special files */
short st_nlink;     /* Number of links */
uid_t st_uid;       /* User ID of the file's owner */
gid_t st_gid;       /* Group ID of the file's group */
off_t st_size;      /* File size in bytes */
timeval_t   st_atimeval; /* Time of last access */
timeval_t   st_mtimeval; /* Time of last data modification */
timeval_t   st_ctimeval; /* Time of last file status change */
                    /* Times measured in microseconds since */
```

```
                        /* 00:00:00 UTC, Jan. 1, 1970 */
#define     st_atime    st_atimeval.tv_sec
                        /* last access time in seconds */
#define     st_mtime    st_mtimeval.tv_sec
                        /* last modify time in seconds */
#define     st_ctime    st_ctimeval.tv_sec
                        /* last change time in seconds */
```

st_mode The mode of the file as described in the **mknod**(2) system call.

st_ino This field uniquely identifies the file in a given filesystem. The pair **st_ino** and **st_dev** uniquely identifies regular files.

st_dev This field uniquely identifies the filesystem that contains the file. Its value may be used as input to the **ustat**(2) system call to determine more information about this filesystem. No other meaning is associated with this value.

st_rdev This field should be used only by administrative commands. It is valid only for block special or character special files and only has meaning on the system where the file was configured.

st_nlink This field should be used only by administrative commands.

st_uid The user ID of the file's owner.

st_gid The group ID of the file's group.

st_size For regular files, this is the address of the end of the file. For pipes or fifos, this is the count of the data currently in the file. For block special or character special, this is not defined.

st_atimeval Time in seconds and microseconds when file data was last accessed. Changed by the following system calls: **creat**(2), **exec**(2), **open**(2), **mkdir**(2), **mknod**(2), **pipe**(2), **utime**(2), and **read**(2).

st_atime Contains the *time_t*(seconds) portion of **st_atimeval.**

st_mtimeval Time in seconds and microseconds when data was last modified. Changed by the following system calls: **creat**(2), **open**(2), **link**(2), **unlink**(2), **mkdir**(2), **rmdir**(2), **rename**(2), **mknod**(2), **pipe**(2), **utime**(2), and **write**(2).

st_mtime Contains the *time_t*(seconds) portion of **st_mtimeval.**

st_ctimeval Time in seconds and microseconds when file status was last changed. Changed by the following system calls: **chmod**(2), **chown**(2), **creat**(2), **open**(2), **link**(2), **mkdir**(2), **rmdir**(2), **rename**(2), **mknod**(2), **pipe**(2), **unlink**(2), **utime**(2), and **write**(2).

st_ctime Contains the *time_t*(seconds) portion of **st_ctimeval.**

stat and **lstat** fail if any of the following are true:

[EACCES] Search permission is denied for a component of the path prefix.

[EFAULT] **buf** or **path** points to an invalid address.

[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**Path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	The named file does not exist.
[ENOTDIR]	A component of the path prefix is not a directory.

fstat fails if any of the following are true:

[EBADF]	**fildes** is not a valid open file descriptor.
[EFAULT]	**buf** points to an invalid address.
[EIO]	An I/O error occurred while reading from or writing to the file system.

SEE ALSO

chmod(2), chown(2), creat(2), link(2), mknod(2), pipe(2), read(2), time(2), unlink(2), utime(2), write(2)

DIAGNOSTICS

Upon successful completion a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

statfs, fstatfs – get filesystem information

SYNOPSIS

```
#include <sys/types.h>
#include <sys/statfs.h>

int statfs (path, buf, len, fstyp)
char *path;
struct statfs *buf;
int len, fstyp;

int fstatfs (fildes, buf, len, fstyp)
int fildes;
struct statfs *buf;
int len, fstyp;
```

DESCRIPTION

statfs returns a *generic superblock* describing a filesystem. It can be used to acquire information about mounted as well as unmounted filesystems, and usage is slightly different in the two cases. In all cases, **buf** is a pointer to a structure (described below) which will be filled by the system call, and **len** is the number of bytes of information which the system should return in the structure. **len** must be no greater than **sizeof (struct statfs)** and ordinarily contains exactly that value; if it holds a smaller value the system fills the structure with that number of bytes. (This allows future versions of the system to grow the structure without invalidating older binary programs.)

If the filesystem of interest is currently mounted, **path** should name a file which resides on that filesystem. In this case the filesystem type is known to the operating system and the **fstyp** argument must be zero. For an unmounted filesystem **path** must name the block special file containing it and **fstyp** must contain the (non-zero) filesystem type. In both cases read, write, or execute permission of the named file is not required, but all directories listed in the pathname leading to the file must be searchable.

The **statfs** structure pointed to by **buf** includes the following members:

```
short  f_fstyp;   /* Filesystem type */
long   f_bsize;   /* Block size */
long   f_frsize;  /* Fragment size */
long   f_blocks;  /* Total number of blocks */
long   f_bfree;   /* Count of free blocks */
long   f_files;   /* Total number of file nodes */
long   f_ffree;   /* Count of free file nodes */
char   f_fname[6];/* Volume name */
char   f_fpack[6];/* Pack name */
long   f_bavail;  /* Block available to non superuser */
```

fstatfs is similar, except that the file named by **path** in **statfs** is instead identified by an open file descriptor **filedes** obtained from a successful **open**(2), **creat**(2), **dup**(2), **fcntl**(2), or **pipe**(2) system call.

statfs obsoletes **ustat**(2) and should be used in preference to it in new programs.

statfs and **fstatfs** fails if one or more of the following are true:

[EACCES]	Search permission is denied for a component of the path prefix.
[EBADF]	**fildes** is not a valid open file descriptor.
[EFAULT]	**buf** or **path** points to an invalid address.
[EINVAL]	**fstyp** is an invalid filesystem type; **path** is not a block special file and **fstyp** is non-zero; **len** is negative or is greater than **sizeof (struct statfs)**.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	The named file does not exist. [ENOTDIR] A component of the path prefix is not a directory.

DIAGNOSTICS

Upon successful completion a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

SEE ALSO

chmod(2), chown(2), creat(2), link(2), mknod(2), pipe(2), read(2), time(2), unlink(2), utime(2), write(2), fs(4)

WARNING

Certain fields of the **statfs** structure may refer to attributes of the underlying file system that are either unknown or undefined. For example, certain fields of the **statfs** structure may be undefined or unknown for remote file systems.

Any unknown or undefined numerical fields of the **statfs** structure are returned as –1. Any unknown or undefined character fields are returned with all characters set to zero. Application programs must be prepared to handle such returns.

NAME

stime – set time

SYNOPSIS

```
int stime (tp)
long *tp;
```

DESCRIPTION

stime sets the system's idea of the time and date. **tp** points to the value of time as measured in seconds from 00:00:00 UTC January 1, 1970.

[EPERM] **stime** fails if the effective user ID of the calling process is not superuser.

SEE ALSO

time(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and is set to indicate the error.

NAME

stty, gtty – set and get terminal state (defunct)

SYNOPSIS

```
#include <sgtty.h>

stty(fd, buf)
int fd;
struct sgttyb *buf;

gtty(fd, buf)
int fd;
struct sgttyb *buf;
```

DESCRIPTION

This interface is obsoleted by **ioctl**(2).

stty sets the state of the terminal associated with **fd**. **gtty** retrieves the state of the terminal associated with **fd**. To set the state of a terminal the call must have write permission.

The **stty** call is actually **ioctl(fd, TIOCSETP, buf)**, while the **gtty** call is **ioctl(fd, TIOCGETP, buf)**. See **ioctl**(2) and **tty**(4) for an explanation.

DIAGNOSTICS

If the call is successful 0 is returned, otherwise –1 is returned and the global variable **errno** contains the reason for the failure.

SEE ALSO

ioctl(2), tty(4)

NAME

sync – update superblock

SYNOPSIS

void sync ( )

DESCRIPTION

sync causes all information in memory that should be on disk to be written out. This includes modified superblocks, modified inodes, and delayed block I/O.

It should be used by programs which examine a filesystem, for example **fsck**, **df**, etc. It is mandatory before a re-boot.

The writing, although scheduled, is not necessarily complete upon return from **sync**.

NAME

syscall – indirect system call

SYNOPSIS

syscall(number, arg, ...)

DESCRIPTION

syscall performs the system call whose assembly language interface has the specified **number** and arguments **arg**.

The r0 value of the system call is returned.

DIAGNOSTICS

When the C-bit is set, **syscall** returns –1 and sets the external variable **errno** (see **intro**(2)).

BUGS

Certain system calls, such as **brk**(2), **dup**(2), **fork**(2), **pipe**(2), **signal**(2), **sigset**(2), **sigaction**(2), **vfork**(2), **wait**(2), and **waitpid**(2), cannot be accessed using this interface. These **syscalls** require extra interface code, which is normally provided in the C library entry points. This interface code is **not** provided through the **syscall**(2) invocation.

Some **syscall** numbers differ between DYNIX/ptx and the i386 version of System V. Programs that do not use the symbolic definitions from *<syscall.h>* may fail when ported to and from other operating systems.

NAME

sysconf – get configurable system variables

SYNOPSIS

#include <unistd.h>

long sysconf (name)
int name;

DESCRIPTION

The **sysconf** function provides a method for the application to determine the current value of a configurable system limit or option (variable).

The **name** argument represents the system variable to be queried. The following system variables are supported:

Configurable Variables

Variable	*name* **Value**
ARG_MAX	
CHILD_MAX	_SC_CHILD_MAX
CLK_TCK	
NGROUPS_MAX	_SC_NGROUPS_MAX
OPEN_MAX	_SC_OPEN_MAX
_POSIX_JOB_CONTROL	_SC_JOB_CONTROL
_POSIX_SAVED_IDS	_SC_SAVED_IDS
_POSIX_VERSION	_SC_VERSION

These variables come from *<limits.h>* or *<unistd.h>;* or from *<time.h>* , and the symbolic constants, defined in *<unistd.h>,* that are the corresponding values used for **name.**

The value of CLK_TCK may be evaluated at run-time. The value returned by **sysconf** for _SC_CLK_TCK is the same as that returned by CLK_TCK.

DIAGNOSTICS

If *name* is an invalid value, **sysconf** returns -1. If the variable corresponding to *name* is not defined on the system, **sysconf** returns -1 without changing the value of **errno.**

Otherwise, **sysconf** returns the current variable value in the system. The value returned is not more restrictive than the corresponding value described to the application when it was complied with *<limits.h>* or *<unistd.h>*. The value does not change during the lifetime of the calling process.

If the following condition occurs, **sysconf** returns -1 and sets **errno** to the corresponding value:

[EINVAL] The value of the argument is invalid.

NAME

sysi86 – machine specific functions

SYNOPSIS

#include <sys/sysi86.h>

int sysi86 (cmd, arg1, arg2, arg3)
int cmd, arg1, arg2, arg3;

DESCRIPTION

sysi86 implements machine specific functions. The **cmd** argument determines the function performed. The number of arguments expected is dependent on the function. The values of the **cmd** arguments are explained in the following paragraphs.

SI86FPHW

When **cmd** is **SI86FPHW**, an indication of the floating-point hardware available is returned. One argument, the address of an integer, is expected. On return from the system call, this integer contains the following values (defined in *<sys/fp.h>*):

FP_387 indicates that the 80387 chip is present

WEITEK_HW indicates that the WTL 1167 is present

SETNAME

This command, which is only available to the superuser, expects an argument of type **char*** which points to a NULL terminated string of at most 7 characters. This command changes the running system's **sysname** and **nodename** (see **uname**(2)) to this string.

SI86MEM

This command returns the size of total memory in bytes.

SI86SWPI

When **cmd** is SI86SWPI, individual swapping areas may be added or the current areas determined. The address of an appropriately primed swap buffer is passed as the only argument. (Refer to the *<sys/swap.h >* header file for details of loading the buffer.)

The format of the swap buffer is:

```
struct swapint {
   char    si_cmd;      /*command: list, add, delete*/
   char    *si_buf;     /*swap file path pointer*/
   int     si_swplo;    /*start block (not used)*/
   int     si_nblks;    /*swap size*/
}
```

Note that the add option of the command may only be exercised by the superuser. If **si_nblks** is 0, the entire device is used. **si_swplo** is not used; it is assumed to be always 0.

If this command succeeds, it returns 0 to the calling process. This command fails, returning -1, if any of the following is true:

[EEXIST] Swap area specified has already been added

[EFAULT] *swapbuf* points to an invalid address

[EFAULT]	*swapbuf.si_buf* points to an invalid address
[EINVAL]	Bad arguments
[ENOSPC]	Too many swap areas in use (if adding)
[ENOTBLK]	Swap area specified is not a block special device

STIME

When **cmd** is STIME, an argument of type **time_t** is expected. This function sets the system time and date. The argument contains the time as measured in seconds from 00:00:00 UTC (Universal Time Code) January 1, 1970. Note that this command is only available to the superuser.

SEE ALSO

swap(1M), uname(2)

DIAGNOSTICS

Upon successful completion, the value of zero is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error. When **cmd** is invalid, **errno** is set to EINVAL on return.

NAME

tcgetattr, tcsetattr - get/set terminal state

SYNOPSIS

```
#include <termios.h>

int tcgetattr (fildes, termios_p)
int fildes;
struct termios *termios_p;

int tcsetattr (fildes, optional_actions, termios_p)
int fildes, optional_actions;
struct termios *termios_p;
```

DESCRIPTION

The **tcgetattr**() function gets the parameters associated with the terminal referred to by *fildes* and stores them in the *termios* structure referrenced by *termios_p*. This function is allowed from a background process; however, the terminal attributes may be subsequently changed by a foreground process.

The **tcsetattr**() function sets the parameters associated with the terminal from the *termios* structure referrenced by *termios_p* as follows:

(1) If *optional_actions* is **tcsanow**() , the change shall occur immediately.

(2) If *optional_actions* is **tcsadrain**() , the change shall occur after all output written to *fildes* has been transmitted. this function should be used when changing parameters that affect output.

(3) If *optional_actions* is **tcsaflush**() , the change shall occur after all output written to the object referred to by *fildes* has been transmitted, and all input that has been received but not read shall be descarded before the change is made.

The symbolic constants for the values of *optional_actions* are defined in **<termios.h>**

Upon successful completion, **tcgetattr**() and **tcsetattr**() will return a value of zero. Otherwise, a value of –1 is returned and *errno* is set to indicate the error.

If any of the following conditions occur, the **tcgetattr**() function shall return -1 and set *errno* to the corresponding value:

[EBADF]
The *fildes* argument is not a valid file descriptor.

[ENOTTY]
The file associated with *fildes* is not a terminal.

If any of the followin conditions occur, the **tcsetattr**() function shall return -1 and set *errno* to the corresponding value:

[EBADF]
The *fildes* argument is not a valid file descriptor.

[EINVAL]
Either *arg* or the data it references is not a proper value.

[ENOTTY]
The file associated with *fildes* is not a terminal.

DIAGNOSTICS

The **tcsetattr**() function is restricted from use by processes belonging to a background process group. Attempts to perform this operation shall cause the process group to be sent a SIGTTOU signal. If the calling process is blocking or ignoring SIGTTOU signals, the process is allowed to perform the operation and the SIGTTOU signal is not sent. See the Terminal Access Control section of the **termio(7)** man page for more details.

NOTES

The **tcgetattr**() function performs the TCGETP ioctl command.

Regarding the **tcsetattr**() function: If *optional_actions* is TCSANOW, the function performs the TCSETP command. If *optional_actions* is TCSADRAIN, the function performs the TCSETPW command. If *optional_actions* is TCSAFLUSH, the function performs the TCSETPF command.

NAME

tcgetpgrp – get foreground process group ID

SYNOPSIS

```
#include <sys/types.h>

pid_t tcgetpgrp (fildes)
int fildes;
```

DESCRIPTION

The **tcgetpgrp** function returns the value of the process group ID of the foreground process group associated with the terminal.

The **tcgetpgrp** function is allowed from a process that is a member of a background-process group; however, the information may be subsequently changed by a process that is a member of a foreground-process group.

On successful completion, **tcgetpgrp** returns the process group ID of the foreground process group associated with the terminal. Otherwise, a value of -1 is returned and **errno** is set to indicate the error.

If any of the following conditions occur, **tcgetpgrp** returns -1 and sets **errno** to the corresponding value:

[EBADF] The **fildes** argument is not a valid file descriptor.

[ENOTTY] The calling process does not have a controlling terminal or the file is not the controlling terminal.

SEE ALSO

setsid(2), setpgid(2), tcsetpgrp(2)

NAME

tcsetpgrp – set foreground process group ID

SYNOPSIS

```
#include <sys/types.h>

int tcsetpgrp (fildes,pgrp_id)
int fildes;
pid_t pgrp_id;
```

DESCRIPTION

If the process has a controlling terminal, **tcsetpgrp** sets the foreground process group ID associated with the terminal to **pgrp_id**. The file associated with **fildes** must be the controlling terminal of the calling process and the controlling terminal must be currently associated with the session of the calling process. The value of **pgrp_id** must match a process group ID of a process in the same session as the calling process.

Upon successful completion, **tcsetpgrp** returns a value of zero. Otherwise, a value of -1 is returned and **errno** is set to indicate the error.

If any of the following conditions occur, **tcsetpgrp** returns -1 and sets **errno** to the corresponding value:

[EBADF] The **fildes** argument is not a valid file descriptor.

[ENOTTY] The calling process does not have a controlling terminal, or the file is not the controlling terminal, or the controlling terminal is no longer associated with the session of the calling process.

[EPERM] The value of **pgrp_id** is within range , but does not match the process group ID of a process in the same session as the calling process.

NAME

time – get time

SYNOPSIS

```
#include <sys/types.h>

time_t time (tloc)
time_t *tloc;
```

DESCRIPTION

time returns the value of time in seconds since 00:00:00 UTC, January 1, 1970.

If **tloc** is non-zero, the return value is also stored in the location to which **tloc** points.

SEE ALSO

stime(2)

WARNINGS

time fails and its actions are undefined if **tloc** points to an illegal address.

DIAGNOSTICS

Upon successful completion, **time** returns the value of time. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

times – get process and child process times

SYNOPSIS

```
#include <sys/types.h>
#include <sys/times.h>

clock_t times (buffer)
struct tms *buffer;
```

DESCRIPTION

times fills the structure pointed to by **buffer** with time-accounting information. The following are the contents of this structure:

```
struct tms {
        clock_t         tms_utime;
        clock_t         tms_stime;
        clock_t         tms_cutime;
        clock_t         tms_cstime;
};
```

This information comes from the calling process and each of its terminated child processes for which it has executed a **wait**. All times are reported in clock ticks per second. Clock ticks are a system-dependent parameter. The specific value for an implementation is defined by the variable **HZ**, found in the include file *<sys/param.h>*.

tms_utime is the CPU time used while executing instructions in the user space of the calling process.

tms_stime is the CPU time used by the system on behalf of the calling process.

tms_cutime is the sum of the **tms_utimes** and **tms_cutimes** of the child processes.

tms_cstime is the sum of the **tms_stimes** and **tms_cstimes** of the child processes.

SEE ALSO

exec(2), fork(2), time(1), time(2), wait(2)

DIAGNOSTICS

Upon successful completion, **times** returns the elapsed real time, in clock ticks per second, from an arbitrary point in the past (e.g., system start-up time). This point does not change from one invocation of **times** to another. If **times** fails, a –1 is returned and **errno** is set to indicate the error.

[EFAULT] **times** fails if **buffer** points to an illegal address.

On Sequent systems, clock ticks occur 100 times per second.

NAME

uadmin – administrative control

SYNOPSIS

```
#include <sys/boot.h>
#include <sys/uadmin.h>

int uadmin (cmd, fcn, mdep)
int cmd, fcn, mdep;
```

DESCRIPTION

uadmin provides control for basic administrative functions. This system call is tightly coupled to the system administrative procedures and is not intended for general use.

As specified by **cmd**, the following commands are available:

A_REBOOT — The system stops immediately without any further processing. The action to be taken next is specified by **fcn**, described below.

A_SHUTDOWN — The system is shutdown. All user processes are killed, the buffer cache is flushed, and the root file system is unmounted. The action to be taken after the system has been shut down is specified by **fcn**.

- AD_HALT — Halt the system.
- AD_BOOT — Reboot the system, using /unix.
- AD_IBOOT — Interactive reboot; user is prompted for system name.
- AD_BOOT_NOKILL — Same as A_SHUTDOWN/AD_BOOT, but don't kill user processes.

The **mdep** argument is a bitmask of options that affects the actions as follows.

- RB_AUTOBOOT — This is the default; the system is rebooted according to the firmware parameters.
- RB_HALT — Halt the system.
- RB_ASKNAME — Interactive reboot; the bootstrap program prompts for the filename to be rebooted.
- RB_NOSYNC — Don't flush the disk buffer cache out to disk before shutting down.

uadmin fails if any of the following are true:

[EPERM] — The effective user ID is not superuser.

DIAGNOSTICS

Upon successful completion, the value returned depends on **cmd** as follows:

A_SHUTDOWN Never returns.

A_REBOOT Never returns.

Otherwise, a value of -1 is returned and **errno** is set to indicate the error.

NAME

ulimit – get and set user limits

SYNOPSIS

long ulimit (cmd, newlimit)
int cmd;
long newlimit;

DESCRIPTION

This function provides for control over process limits. The **cmd** values available follow. If the value of command is 1, 3, or 4 below (get a current limit), **newlimit** is ignored.

1. Get the regular file size limit of the process. The limit is in units of 512-byte blocks and is inherited by child processes. Files of any size can be read.
2. Set the regular file size limit of the process to the value of **newlimit**. Any process may decrease this limit, but only a process with an effective user ID of superuser may increase the limit. **ulimit** fails and the limit is unchanged if a process with an effective user ID other than superuser attempts to increase its regular file size limit. [EPERM]
3. Get the maximum possible break value [see **brk**(2)].
4. Get the current value of the maximum number of open files per process configured in the system.

ulimit fails if any of the following occur:

[EPERM] The caller was not superuser, and tried to increase it's file size limit.

[EINVAL] An unsupported value for the **cmd** argument was used.

SEE ALSO

brk(2), write(2)

WARNINGS

ulimit is effective in limiting the growth of regular files. Pipes are currently limited to 8,192 bytes.

DIAGNOSTICS

Upon successful completion, a non-negative value is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

umask – set and get file creation mask

SYNOPSIS

```
#include <sys/types.h>
mode_t umask (cmask)
mode_t cmask;
```

DESCRIPTION

umask sets the process's file mode creation mask to **cmask** and returns the previous value of the mask. Only the low-order 9 bits of **cmask** and the file mode creation mask are used.

SEE ALSO

mkdir(1), sh(1), chmod(2), creat(2), mknod(2), open(2)

DIAGNOSTICS

The previous value of the file mode creation mask is returned.

NAME

umount – unmount a filesystem

SYNOPSIS

int umount (file)
char *file;

DESCRIPTION

umount requests that a previously mounted filesystem contained on the block special device or directory identified by **file** be unmounted. **file** is a pointer to a pathname. After unmounting the filesystem, the directory upon which the filesystem was mounted reverts to its ordinary interpretation.

umount may be invoked only by the superuser.

umount fails if one or more of the following are true:

[EBUSY]	A file on **file** is busy.
[EFAULT]	**file** points to an illegal address.
[EINVAL]	**file** does not exist.
[EINVAL]	**file** is not mounted.
[ENOTBLK]	**file** is not a block special device.
[EPERM]	The process's effective user ID is not superuser.

SEE ALSO

mount(2)

DIAGNOSTICS

Upon successful completion a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

uname – get name of current operating system

SYNOPSIS

#include <sys/utsname.h>

int uname (name)
struct utsname *name;

DESCRIPTION

uname stores information identifying the current operating system in the structure pointed to by **name.**

uname uses the structure defined in *<sys/utsname.h>* whose members are:

```
char   sysname[9];
char   nodename[9];
char   release[9];
char   version[9];
char   machine[9];
```

uname returns a null-terminated character string naming the current operating system in the character array **sysname.** Similarly, **nodename** contains the name that the system is known by on a communications network. **release** and **version** further identify the operating system. **machine** contains a standard name that identifies the hardware that the operating system is running on.

[EFAULT] **uname** fails if **name** points to an invalid address.

SEE ALSO

uname(1)

DIAGNOSTICS

Upon successful completion, a non-negative value is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

unlink – remove directory entry

SYNOPSIS

int unlink (path)
char *path;

DESCRIPTION

unlink removes the directory entry named by the path name pointed to by **path.**

The named file is unlinked unless one or more of the following are true:

[EACCES]	Search permission is denied for a component of the path prefix.
[EACCES]	Write permission is denied on the directory containing the link to be removed.
[EACCES]	The parent directory has the sticky bit set and the file is not writable by the user and the user does not own the parent directory and the user does not own the file and the user is not superuser
[EBUSY]	The entry to be unlinked is the mount point for a mounted file system.
[EFAULT]	**path** points outside the process's allocated address space.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.
[ENOENT]	The named file does not exist.
[ENOTDIR]	A component of the path prefix is not a directory.
[EPERM]	The named file is a directory and the effective user ID of the process is not superuser.
[EROFS]	The directory entry to be unlinked is part of a read-only file system.
[ETXTBSY]	The entry to be unlinked is the last link to a pure procedure (shared text) file that is being executed.

When all links to a file have been removed and no process has the file open, the space occupied by the file is freed and the file ceases to exist. If one or more processes have the file open when the last link is removed, the removal is postponed until all references to the file have been closed.

SEE ALSO

rm(1), close(2), link(2), open(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error.

NAME

ustat – get filesystem statistics

SYNOPSIS

```
#include <sys/types.h>
#include <ustat.h>

int ustat (dev, buf)
dev_t dev;
struct ustat *buf;
```

DESCRIPTION

ustat returns information about a mounted filesystem. **dev** is a device number identifying a device containing a mounted filesystem. **buf** is a pointer to a **ustat** structure that includes the following elements:

```
daddr_t  f_tfree;     /* Total free blocks */
ino_t    f_tinode;    /* Number of free inodes */
char     f_fname[6];  /* Filsys name */
char     f_fpack[6];  /* Filsys pack name */
```

ustat fails if one or more of the following are true:

[EFAULT] **buf** points outside the process's allocated address space.

[EINVAL] **dev** is not the device number of a device containing a mounted filesystem.

SEE ALSO

stat(2), fs(4)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

WARNING

Certain fields of the **ustat** structure may refer to attributes of the underlying file system that are either unknown or undefined. For example, certain fields of the **ustat** structure may be undefined or unknown for remote file systems.

Any unknown or undefined numerical fields of the **ustat** structure are returned as –1. Any unknown or undefined character fields are returned with all characters set to zero. Application programs must be prepared to handle such returns.

NAME

utime – set file access and modification times

SYNOPSIS

```
#include <sys/types.h>
int utime (path, times)
char *path;
struct utimbuf *times;
```

DESCRIPTION

path points to a path name naming a file. **utime** sets the access and modification times of the named file.

If **times** is NULL, the access and modification times of the file are set to the current time. A process must be the owner of the file or have write permission to use **utime** in this manner.

If **times** is not NULL, **times** is interpreted as a pointer to a **utimbuf** structure and the access and modification times are set to the values contained in the designated structure. Only the owner of the file or the superuser may use **utime** this way.

The times in the following structure are measured in seconds since 00:00:00 UTC, Jan. 1, 1970.

```
struct   utimbuf       {
   time_t actime;      /* access time */
   time_t modtime;     /* modification time */
};
```

utime fails if one or more of the following are true:

[EACCES]	Search permission is denied by a component of the path prefix.
[EACCES]	The effective user ID is not superuser and not the owner of the file and **times** is NULL and write access is denied.
[EFAULT]	**times** is not NULL and points outside the process's allocated address space.
[EFAULT]	**path** points outside the process's allocated address space.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating **path.**
[ENAMETOOLONG]	**path** was longer than PATH_MAX characters.
[ENAMETOOLONG]	**path** referenced a symbolic link which, when expanded, resulted in a path name longer than PATH_MAX characters.
[ENAMETOOLONG]	A component of **path** was longer than NAME_MAX characters.

[ENOENT]	The named file does not exist.
[ENOTDIR]	A component of the path prefix is not a directory.
[EPERM]	The effective user ID is not superuser and not the owner of the file and **times** is not NULL.
[EROFS]	The filesystem containing the file is mounted read-only.

SEE ALSO

stat(2)

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

vfork – spawn new process in a virtual memory efficient way

SYNOPSIS

#include <sys/types.h>
pid_t vfork()

DESCRIPTION

vfork can be used to create new processes without fully copying the address space of the old process, which is horrendously inefficient in a paged environment. It is useful when the purpose of **fork**(2) would have been to create a new system context for an **exec**. **vfork** differs from **fork** in that the child borrows the parent's memory and thread of control until a call to **exec**(2) or an exit (either by a call to **exit**(2) or abnormally.) The parent process is suspended while the child is using its resources.

vfork returns 0 in the child's context and (later) the **pid** of the child in the parent's context.

vfork can normally be used just like **fork**. It does not work, however, to return while running in the child's context from the procedure which called **vfork** since the eventual return from **vfork** would then return to a no longer existent stack frame. Be careful, also, to call **_exit** rather than **exit** if you can't *exec*, since **exit** flushes and closes standard I/O channels, and thereby messes up the parent process's standard I/O data structures. (Even with **fork** it is wrong to call **exit** since buffered data would then be flushed twice.)

SEE ALSO

fork(2), exec(2), wait(2)

DIAGNOSTICS

Same as for **fork**.

BUGS

Users should not depend on the memory sharing semantics of **vfork** as it is an artifact of the implementation.

To avoid a possible deadlock situation, processes which are children in the middle of a **vfork** are never sent SIGTTOU or SIGTTIN signals; rather, output or **ioctl**s are allowed and input attempts result in an end-of-file indication.

NAME

wait, waitpid – wait for a child process to stop or terminate

SYNOPSIS

```
#include <sys/wait.h>
#include <sys/types.h>

pid_t wait (stat_loc)
int *stat_loc;

pid_t waitpid (pid, stat_loc, options)
pid_t  pid;
int *stat_loc;
int    options;
```

DESCRIPTION

wait and **waitpid** allow the calling process to obtain status information pertaining to one of its child processes. Various options permit status information to be obtained for child processes that have terminated or stopped. If status information is available for two or more child processes, the order in which their status is reported is unspecified.

The **wait** function suspends execution of the calling process until status information for one of its terminated child processes is available, or until delivery of a signal whose action is either to execute a signal-catching function or to terminate the process. If status information is available prior to the call to **wait**, it returns immediately.

The **waitpid** function behaves identically to the **wait** function, if the **pid** argument has the value –1 and the **options** argument has a value of zero. Otherwise, its behavior is dependent on the values of **pid** and **options**.

The **pid** argument specifies a set of child processes for which status is requested. The **waitpid** function shall only return the status of a child process from this set.

(1) If **pid** is equal to –1, status is requested for any child process. In this respect, **waitpid()** is then equivalent to **wait()**.
(2) If **pid** is greater than zero, it specifies the process ID of a single child process for which status is requested.
(3) If **pid** is equal to zero, status is requested for any child process whose process group ID is equal to that of the calling process.
(4) If **pid** is less than –1, status is requested for any child process whose process group ID is equal to the absolute value of **pid**.

The **options** argument is constructed the the bitwise inclusive OR of zero or more of the following flags, defined in the header *<sys/wait.h>*:

WNOHANG The **waitpid()** function shall not suspend execution of the calling process if status is not immediately available for one of the child processes specified by **pid**.

WUNTRACED The status of any child process specified by **pid** that are stopped, and whose status has not yet been reported since they stopped, are also reported.

If **stat_loc** is non-zero, the requested information called **status** are stored in the location pointed to by **stat_loc**. **status** can be used to differentiate

between stopped and terminated child processes and if the child process terminated, **status** identifies the cause of termination and passes useful information to the parent. This is accomplished by using the following macros, that are defined in *<sys/wait.h>* and evaluate to integral expressions; the **stat_val** argument is the integer value pointed to by **stat_loc**.

WIFEXITED(stat_val)
: Evaluates to a non-zero value if status was returned for a child process that terminated normally.

WEXITSTATUS(stat_val)
: If the value of WIFEXITED(stat_val) is non-zero, this macro evaluates to the low-order 8 bits of the **status** argument that the child process passed to **_exit()** or **exit()**, or the value the child process returned from **main()**.

WIFSIGNALED(stat_val)
: Evaluates to a non-zero value if the status was returned for a child process that terminated due to the receipt of a signal that was not caught.

WTERMSIG(stat_val)
: If the value of WIFSIGNALED(stat_val) is non-zero, this macro evaluates to the number of the signal the caused the termination of the child process.

WIFSTOPPED(stat_val)
: Evaluates to a non-zero value if status was returned for a child process that is currently stopped.

WSTOPSIG(stat_val)
: If the value of WIFSTOPPED(stat_val) is non-zero, this macro evaluates to the number of the signal the caused the child process to stop.

WIFCORESIG(stat_val)
: Evaluates to a non-zero value if the status was returned for a child process that terminated due to the receipt of a signal that was not caught and caused a core dump.

If a parent process terminates without waiting for its child processes to terminate, the parent process ID of each child process is set to 1. This means the initialization process inherits the child processes (see **intro**(2)).

wait fails and returns immediately if any of the following are true:

[ECHILD] The calling process has no existing unwaited-for child processes and (for waitpid) if the process group specified by **pid** does not exist, or is not a child.

[EINTR] **wait** returned due to the receipt of a signal.

[EINVAL] if the value of the **options** argument is not valid.

SEE ALSO

exec(2), exit(2), fork(2), intro(2), pause(2), ptrace(2), signal(2)

WARNINGS

wait fails and its actions are undefined if **stat_loc** points to an invalid address.

DIAGNOSTICS

If **wait** returns due to the receipt of a signal, a value of –1 is returned to the calling process and **errno** is set to EINTR. If **wait** returns due to a stopped or terminated child process, the process ID of the child is returned to the calling process. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

NAME

write – write on a file

SYNOPSIS

int write (fildes, buf, nbyte)
int fildes;
char *buf;
unsigned nbyte;

DESCRIPTION

fildes is a file descriptor obtained from a **creat**(2), **open**(2), **dup**(2), **fcntl**(2), or **pipe**(2) system call.

write attempts to write **nbyte** bytes from the buffer pointed to by **buf** to the file associated with the **fildes**.

On devices capable of seeking, the actual writing of data proceeds from the position in the file indicated by the file pointer. Upon return from **write**, the file pointer is incremented by the number of bytes actually written.

On devices incapable of seeking, writing always takes place starting at the current position. The value of a file pointer associated with such a device is undefined.

If the O_APPEND flag of the file status flags is set, the file pointer is set to the end of the file prior to each write.

For regular files, if the O_SYNC flag of the file status flags is set, the write does not return until both the file data and file status have been physically updated. This function is for special applications that require extra reliability at the cost of performance. For block special files, if O_SYNC is set, the write does not return until the data has been physically updated.

A write to a regular file will be blocked if mandatory file/record locking is set (see **chmod**(2)), and there is a record lock owned by another process on the segment of the file to be written. If O_NONBLOCK is not set, the write sleeps until the blocking record lock is removed.

For STREAMS (see **intro**(2)] files, the operation of **write** is determined by the values of the minimum and maximum **nbyte** range (packet size) accepted by the stream. These values are contained in the topmost stream module. Unless the user pushes (see I_PUSH in **streamio**(7)) the topmost module, these values can not be set or tested from user level. If **nbyte** falls within the packet size range, **nbyte** bytes are written. If **nbyte** does not fall within the range and the minimum packet size value is zero, **write** breaks the buffer into maximum packet size segments prior to sending the data downstream (the last segment may contain less than the maximum packet size). If **nbyte** does not fall within the range and the minimum value is non-zero, **write** fails with **errno** set to ERANGE. Writing a zero-length buffer (**nbyte** is zero) sends zero bytes with zero returned.

For STREAMS files, if O_NONBLOCK is not set and the stream cannot accept data (the stream write queue is full due to internal flow control conditions), **write** blocks until data can be accepted. O_NDELAY prevents a process from blocking due to flow control conditions. If O_NONBLOCK is set and the stream can not accept data, **write** fails. If O_NONBLOCK is set and part of the buffer has been written when a condition in which the stream can not accept

additional data occurs, **write** terminates and returns the number of bytes written.

write fails and the file pointer remains unchanged if any of the following are true:

[EAGAIN] Mandatory file/record locking was set, O_NONBLOCK was set, and there was a blocking record lock.

[EAGAIN] Total amount of system memory available when reading via raw I/O is temporarily insufficient.

[EAGAIN] Attempt to write to a stream that can not accept data with the O_NONBLOCK flag set.

[EBADF] **fildes** is not a valid file descriptor open for writing.

[EDEADLK] The write was going to go to sleep and cause a deadlock situation to occur.

[EFAULT] **buf** points outside the process's allocated address space.

[EFBIG] An attempt was made to write a file that exceeds the process's file size limit or the maximum file size (see **ulimit**(2)).

[EINTR] A signal was caught during the **write** system call.

[EINVAL] Attempt to write to a stream linked below a multiplexor.

[ENOLCK] The system record lock table was full, so the write could not go to sleep until the blocking record lock was removed.

[ENOSPC] During a **write** to an ordinary file, there is no free space left on the device.

[EIO] A hangup occurred on the stream being written to.

[EIO] An I/O error occurred while reading from or writing to the file system.

[EIO] The process is in a background process group and is attempting to write to its controlling terminal. TGSTOP is set, the process is neither ignoring nor blocking SIGTTOU signals, and the process group of the process is orphaned.

[EPIPE and SIGPIPE signal] An attempt is made to write to a pipe that is not open for reading by any process.

[ERANGE] Attempt to write to a stream with **nbyte** outside specified minimum and maximum write range, and the minimum value is non-zero.

[ENOMEM] The size of the write request to a raw device would have required the process to exceed its maximum resident set size.

If a **write** requests that more bytes be written than there is room for (e.g., the **ulimit** (see **ulimit**(2)) or the physical end of a medium), only as many bytes as there is room for are written.

For example, suppose there is space for 20 bytes more in a file before reaching a limit. A write of 512-bytes returns 20. The next write of a non-zero number of bytes gives a failure return (except as noted below).

Write requests to a pipe (or FIFO) of PIPE_BUF bytes or less will not be interleaved with data from other processes doing writes on the same pipe. Writes of greater than PIPE_BUF bytes may have data interleaved, on arbitrary boundaries, with writes by other processes to that same pipe.

If the pipe being written has the O_NONBLOCK flag of the file flag word set, and the number of bytes in the transfer is PIPE_BUF bytes or less, a write to a pipe which hasn't buffer space available to contain the entire transfer request returns a count of -1 and sets **errno** to EAGAIN. If the number of bytes in the write request is greater than PIPE_BUF and O_NONBLOCK is set, the write may transfer as many bytes as can fit into the pipe at that time, and return the number of bytes actually transferred.

If O_NONBLOCK is clear, **writes** to a "full" pipe (or FIFO) block until buffer space becomes available in that pipe. If the number of bytes requested to transfer is greater than PIPE_BUF, then the transfer may be done in several pieces as buffer space in that pipe becomes available. Under these circumstances, data from other processes writing to that pipe may be interleaved in the pipe with the data from this transfer. If the number of bytes requested is PIPE_BUF or less, the process blocks until space is available in the pipe for the entire transfer. When buffer space becomes available in that pipe, the data is transferred atomically into that pipe's buffer.

A write to a STREAMS file can fail if an error message has been received at the stream head. In this case, **errno** is set to the value included in the error message.

SEE ALSO

intro(2), creat(2), dup(2), fcntl(2), intro(2), lseek(2), open(2), pipe(2), ulimit(2)

DIAGNOSTICS

Upon successful completion, the number of bytes actually written is returned. Otherwise, –1 is returned and **errno** is set to indicate the error.

NAME

intro – introduction to Sequent system calls

SYNOPSIS

#include <errno.h>

DESCRIPTION

This section describes the Sequent-specific system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value. This is almost always –1 or the NULL pointer; the individual descriptions specify the details. An error number is also made available in the external variable **errno. errno** is not cleared on successful calls, so it should be tested only after an error has been indicated.

Each system call description attempts to list all possible errors. Refer to the introduction to Section 2 for a complete list of the error numbers and their names as defined in *<errno.h>*, and for more information about system calls.

SEE ALSO

intro(2)

NAME

bdflush – update daemon

SYNOPSIS

bdflush (autoup, bdflushrate)
int autoup;
int bdflushrate;

DESCRIPTION

bdflush starts a daemon that flushes modified disk buffers out to disk. Only one daemon can be active at any given time.

autoup is the maximum age of delayed write buffers in seconds before they get written out. **bdflushrate** is the rate at which **bdflush** checks for delayed write buffers.

bdflush fails if any of the following is true:

[EBUSY] If another daemon is already active.

[EINVAL] If either **autoup** or **bdflushrate** is negative.

[EPERM] If the caller is not the superuser.

NOTES

This system call never returns.

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

fvhangup – virtually hangup the specified controlling terminal

SYNOPSIS

fvhangup(line)
char *line;

DESCRIPTION

fvhangup is used by the terminal set-up process **getty**(1a) and some other processes to give users controlled access to terminals at login by revoking access of the previous users' processes to the terminal. To effect this, **fvhangup** searches the system tables for references to the specified *line*, revoking access permissions on each instance of the terminal that it finds. Further attempts to access the terminal by the affected processes yield I/O errors (EIO).

This call is available only to the superuser.

ERRORS

[EPERM]	The caller is not the superuser.
[ENOTTY]	The specified *line* is not a character special device.
[ESRCH]	No file descriptors could be found that are currently using this *line*.

SEE ALSO

getty (1a)

NOTES

Any I/O operation in progress on the *line* may succeed.

Care should be taken to ensure that no new opens can be performed on the device during the call to **fvhangup**, as such opens may not be subject to the effects of **fvhangup**.

This is a Sequent library routine. To use it, specify the **-lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

get_process_stats – get resource utilization information

SYNOPSIS

```
#include <sys/types.h>
#include <sys/procstats.h>

get_process_stats(tv, who, pst, cpst)
timeval_t *tv;
int who;
struct process_stats *pst, *cpst;
```

DESCRIPTION

get_process_stats returns information describing the resources utilized by the process identified by **who** and all its terminated child processes. **who** is either the pid of the desired process, or the value **PS_SELF**, defined in the header file **<sys/procstats.h>**. If **who** is given value **PS_SELF**, the statistics returned are for the calling process. In order to get statistics about the process **who**, the caller must either be superuser, or have the same real or effective uid as the process identified by **who**. **tv** returns the current time of day. If **pst** or **cpst** are non-zero, the buffers they point to are filled in with the following structure:

```
struct process_stats {
  timeval_t ps_utime;  /*user time */
  timeval_t ps_stime;  /*system time */
  ulong ps_maxrss;     /*max resident set size (in HW pages) */
  ulong ps_pagein;     /*page faults that resulted in I/O */
  ulong ps_reclaim;    /*pages reclaimed from free/dirty list */
  ulong ps_zerofill;   /*pages zero filled on demand */
  ulong ps_pffincr;    /*times PFF increased process's RSS */
  ulong ps_pffdecr;    /*times PFF decreased process's RSS */
  ulong ps_swap;       /*times process was swapped */
  ulong ps_syscall;    /*number of system calls */
  ulong ps_volcsw;     /*voluntary context switches */
  ulong ps_involcsw;   /*involuntary context switches */
  ulong ps_signal;     /*signals received */
  ulong ps_lread;      /*filesystem "logical" reads */
  ulong ps_lwrite;     /*filesystem "logical" writes */
  ulong ps_bread;      /*filesystem disk reads */
  ulong ps_bwrite;     /*filesystem disk writes */
  ulong ps_phread;     /*raw device reads */
  ulong ps_phwrite;    /*raw device writes */
};
```

The fields are interpreted as follows:

ps_utime	Total amount of time spent executing in user mode
ps_stime	Total amount of time spent in the system executing on behalf of the process(es)
ps_maxrss	Maximum resident set size utilized in pages (as defined by **getpagesize**(2))

ps_pagein	Number of page faults serviced which required I/O activity
ps_reclaim	Number of page faults serviced without any I/O activity; here I/O activity is avoided by reclaiming a page frame from the list of pages awaiting reallocation
ps_zerofill	Number of pages which are zero-filled on demand
ps_pffincr	Number of times a process's resident set size was increased by the kernel
ps_pffdecr	Number of times a process's resident set size was decreased by the kernel
ps_swap	Number of times a process was swapped out of main memory
ps_syscall	Number of system calls performed
ps_volcsw	Number of times a context switch resulted due to a process voluntarily giving up the processor before its time slice was completed (usually to await availability of a resource)
ps_involcsw	Number of times a context switch resulted due to a higher priority process becoming runnable or because the current process exceeded its time slice
ps_signal	Number of signals received
ps_lread	Number of read system calls performed
ps_lwrite	Number of write system calls performed
ps_bread	Number of reads from disk performed to service file system logical reads
ps_bwrite	Number of writes to disk performed to service file system logical writes
ps_phread	Number of reads to raw disk devices
ps_phwrite	Number of writes to raw disk devices

SEE ALSO

wait(2)

DIAGNOSTICS

In the event of a failure, the system call returns a value of -1, and **errno** is set to one of the values below.

[EFAULT]	The memory pointed to by **tv, cpst,** or **pst** lies outside the valid address space for the process.
[EPERM]	Permission is denied to the information about the process specified by **who.** This means the caller is not superuser, and that the real or effective uid's of the caller do not match the real or effective uid of the process identified by **who.**
[ESRCH]	The process specified by **who** does not exist.
[EAGAIN]	The process specified by **who** has been swapped out.

NOTES

The numbers **ps_bread** and **ps_bwrite** account only for real I/O ; data

supplied by the cacheing mechanism is charged only to the first process to read or write the data.

This is a Sequent library routine. To use it, specify the **-lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

BUGS

Due to the multiprocessing nature of the operating system, it is possible for the CPU time (user + system time) for a child process to exceed the elapsed real time, especially if the child process forks and waits for children of its own.

NAME

getdtablemax – get maximum in-use descriptor table index

SYNOPSIS

int getdtablemax

DESCRIPTION

Each process has a fixed size descriptor table which is guaranteed to have at least OPEN_MAX slots. The entries in the descriptor table are numbered with small integers starting at 0. **getdtablemax** returns the highest in-use index of this table. The system guarantees that descriptor table indices greater than **getdtablemax** and less than **getdtablesize** are not in use by the calling process.

SEE ALSO

intro(2), close(2), dup(2), open(2), getdtablesize(2seq), setdtablesize(2seq)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

getdtablesize – get descriptor table size

SYNOPSIS

int getdtablesize()

DESCRIPTION

Each process has a fixed size descriptor table which is guaranteed to have at least OPEN_MAX slots. The entries in the descriptor table are numbered with small integers starting at 0. **getdtablesize** returns the size of this table.

SEE ALSO

intro(2), close(2), dup(2), open(2), setdtablesize(2seq)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

getpagesize – get system page size

SYNOPSIS

int getpagesize()

DESCRIPTION

getpagesize returns the number of bytes in a page. Page granularity is the granularity of many of the memory management calls.

The page size is a **system** page size and may not be the same as the underlying hardware page size.

SEE ALSO

sbrk(2)

NOTES

This is a Sequent library routine. To use it, specify the **-lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

mmap – map an open file into the process's address space

SYNOPSIS

```
#include <sys/types.h>
#include <sys/mman.h>

mmap(addr, len, prot, share, fd, pos)
caddr_t     addr;       /* starting virt-addr */
int         len;        /* length (bytes) to map */
int         prot;       /* RO, RW encoding */
int         share;      /* private/shared modifications */
int         fd;         /* open file to be mapped */
off_t       pos;        /* where in file to begin map */
```

DESCRIPTION

mmap causes the file referenced by **fd**, starting at byte **pos** for **len** bytes in the file, to be mapped into the calling process's address space, starting at virtual address **addr** for **len** bytes, using protection specified by **prot**; modifications to the mapped memory are either private to the process or shared, as specified by **share**. **mmap** can be used to allocate regions of shared memory, to map files into memory, and to access special regions of the physical address space.

fd must reference an open regular (IFREG) or character special (IFCHR) file. The device driver that implements the IFCHR special file must support mapping for this to succeed. Typically, a regular file is used to map shared memory.

The **share** argument specifies whether modifications to a mapped file are to be kept private to the calling process or shared with other processes accessing the file. If **share** is MAP_SHARED , all modifications to the file are shared with others who have it concurrently mapped. If **share** is MAP_PRIVATE , all modifications are local to the calling process; this doesn't restrict other processes from mapping the file. MAP_SHARED and MAP_PRIVATE do not restrict **read** and **write** system-calls. If **share** is MAP_ZEROFILL , the space indicated from **addr** for **len** bytes is replaced by private pages that are zero-filled when referenced. MAP_ZEROFILL ignores the **fd** argument, and **pos** is ignored other than being checked for alignment (specifying fd and pos = 0 is recommended).

The **prot** argument should be PROT_WRITE for write access to the mapped region, PROT_READ for read access, PROT_EXEC for executable access. These values can be ORed to obtain read-write access, etc. For programming convenience, PROT_RDWR is defined as (PROT_READ | PROT_WRITE). The file access permissions on **fd** must allow the requested access. The **prot** argument affects only the calling process; other processes mapping the same file may have different access.

The **addr, len,** and **pos** arguments must be integral multiples of the system page size, as defined by **getpagesize**(2). It is possible to map over previously mapped pages. The newly created address space cannot overlap with a shared memory region created by the **shmat**(2) system call. If **addr** and **len** specify a nonexistent part of the process's address space, the process's data segment is grown to accommodate the request. The process's break address (see **brk**(2)) is unchanged. Reference to any holes between the mapped region and the rest

of the data segment result in a segmentation fault (SIGSEGV). **mmap** does not allow mapping over text or stack pages.

When memory is mapped to a regular file, the file acts like a paging area for the mapped memory region. **read** and **write** operations to mapped regions of the file also affect the corresponding memory. The memory contents are copied out to the file when the process is swapped or when it exits, or when the region is otherwise unmapped by the last process that has it mapped. For programs that use shared memory but do not need a permanent disk image of the memory, the file associated with **fd** can be unlinked (see **unlink**(2)) even before the call to **mmap**: if the file is unlinked when the region is unmapped, the disk space is not updated. A **close**(2) operation on the **fd** does not unmap the region, that is, the mapping persists. The map is implicitly deleted on **exit**(2) or by explicitly calling **munmap**(2seq).

Regular files have their size rounded up to a file-system block boundary. Any non-existent space in the file at the time of the **mmap** request (for example, in a sparse file) is allocated, and filled with zeros when referenced. Both of these operations require write access to the file.

The type of file referenced may impose further restrictions on the **pos**, **offset**, or other parameters. Refer to the manual entry of the relevant device driver (for example, **pmap**(7)) for details.

mmap can be called multiple times with the same file descriptor, resulting in several (possibly overlapping) mapped regions. A process can have up to NUMMAP regions mapped simultaneously; mappings that are completely overlapped by subsequent mappings are not counted in this total. Mappings which use the same file descriptor, and **addr** and **pos** arguments which align virtually with a previous mapping, also don't count in this total; the simplest case is mapping more of a file, starting from the end of a previous mapping. If a file is closed and reopened between two calls to **mmap**, they are counted as separate maps.

All mapped files remain mapped in both the parent and child process after a **fork**. All **exec** and **exit** system calls, when successful, remove all maps the calling process had established. If a process has any maps, **vfork** behaves exactly like **fork**. A child of a **vfork** that has not yet **exec**ed a new image cannot successfully execute **mmap**.

There are three types of mapping: *paged, physical*, and *non-paged* memory. The type of mapping is determined by the type of file being mapped. Paged maps support shared memories and mapped regular files. Physical maps deal with hardware that has restrictive access capability (for example, the MULTIBUS address space). Non-paged memory maps are typically used for special reserved areas of system memory; they are assumed to behave exactly like memory, supporting accesses of arbitrary size and alignment, DMA , etc.

System services (raw I/O , read/write, stat, etc.) are supported in paged and non-paged memory maps; attempts at such services in physically mapped address space result in an error, typically EFAULT . Core dumps include a copy of any mapped address space; however, physically mapped addresses read as zero.

Regular files (IFREG) are always page-mapped. Character special files (IFCHR) can support paged, physical, or non-paged maps, depending on the underlying

hardware. Physical and non-paged maps are always valid in the process address space; references do not cause a page fault.

When **mmap** increases a program's address space, it also attempts to increase its allowable resident set size.

DIAGNOSTICS

mmap returns 0 when successful. Otherwise it returns -1 and **errno** is set to indicate the error.

EXAMPLES

The following code sets up a 1-Mbyte region of shared memory at the first page boundary above the current program break. This region is shared with the process's children and with any other process that maps the file **shmem**:

```
pgsz = getpagesize();
shm_base = (char *) (((int)sbrk(0) + (pgsz-1)) & ~(pgsz-1));
fd = open ("shmem", O_CREAT | O_RDWR, 0666);
mmap (shm_base, 0x100000, PROT_RDWR, MAP_SHARED, fd, 0);
```

ERRORS

[EACCES] **fd** does not allow the desired access (read or write), or a write-only file descriptor was used.

[EACCES] A mapped regular file must be extended to a file-system block boundary, or the file must have space allocated, and the file descriptor is read-only.

[EFBIG] The **pos** and **len** arguments would create too large a file.

[EINVAL] **addr**, **pos**, or **len** is not a multiple of the system page size.

[EINVAL] **prot** did not specify at least one of PROT_WRITE or PROT_READ; **share** did not specify MAP_SHARED, MAP_PRIVATE, or MAP_ZEROFILL; or **share** specified MAP_ZEROFILL but **prot** did not contain PROT_RDWR.

[EINVAL] **fd** does not represent a regular or character special file.

[EINVAL] The process is the child of a **vfork**.

[EINVAL] The area defined by the **addr** and **len** arguments overlaps text or stack pages of the process, or an area created by **shmat**.

[EINVAL] **mmap**ping a file which has mandatory record locking enabled is not supported.

[EMFILE] The system-defined per-process limit on the number of **mmap**ed files (currently NUMMAP) was exceeded.

[ENFILE] The system-wide limit on the number of mapped regular files was exceeded. This limit is defined by the variable **nmfile** in *param_space.c*.

[ENODEV] The device driver indicated by **fd** does not support mapping.

[ENOMEM] There is no swap space for the page table of a mapped regular file, or you are trying to create too large a process.

[ENOSPC] A mapped regular file was sparse and there was insufficient space in the file-system to satisfy the request.

Other error values may be returned by some device drivers when requested to map. See the relevant driver manual entry for details.

SEE ALSO

munmap(2seq), pmap(7), vm_ctl(seq2), fork(2), exec(2), getpagesize(seq2)

NOTES

Due to a hardware restriction, PROT_WRITE implies PROT_READ also. PROT_EXEC is ignored.

To minimize overhead, mapped regions should be kept as close as possible to the low end of process memory.

Address space holes under the process break read as zeros in core files.

This is a Sequent library routine. To use it, specify the **-lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

BUGS

A mapped file may not be truncated.

If a file is extended to a file-system block boundary, its original size is lost.

Current restrictions on what parts of the address space can be re-mapped should be lifted.

NAME

munmap – unmap part of the process's address space

SYNOPSIS

```
munmap(addr, len)
caddr_t     addr;       /* starting virt-addr */
int         len;        /* length (bytes) to unmap */
```

DESCRIPTION

munmap removes a mapping created by **mmap(2seq)**. The **addr** and **len** arguments must be integral multiples of the system page size, as defined by **getpagesize(2seq)**.

After successful completion of **munmap**, the address space from **addr** through **addr+len**-1 does not exist and will result in a segmentation fault (SIGSEGV) if referenced.

DIAGNOSTICS

Munmap returns zero when successful. Otherwise it returns -1 and **errno** is set to indicate the error.

ERRORS

[EINVAL] **addr** or **len** is not a multiple of the system page size.

[EINVAL] The area defined by **addr** and **len** overlaps the text or stack segment or a non-existent part of the address space.

SEE ALSO

mmap(2seq), vm_ctl(2seq), getpagesize(2seq)

NOTES

The current implementation can unmap only data space.

Any address space holes created by **munmap** read as zeros in a core file.

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

proc_ctl – control over various process attributes

SYNOPSIS

```
#include <sys/time.h>
#include <sys/resource.h>

proc_ctl(cmd, who, arg)
int     cmd;
int     who;
int     arg;
```

DESCRIPTION

This call supports manipulation of various process attributes. The only currently supported **cmd** is PROC_PRIOAGE. **cmd**, **who**, and **arg** are interpreted as follows for **cmd** = PROC_PRIOAGE:

who is a process ID (pid); zero means the calling process. Priority aging of the selected process is enabled or disabled depending on **arg** being non-zero or zero (respectively). When priority aging is disabled, the system no longer takes the calling process's CPU usage history into consideration when selecting among processes to run. The calling process's effective uid must be root or the same as the selected process. Any uid can enable priority aging. A root process, or any process if the system is configured for it, can disable priority aging. The priority aging attribute is inherited across **fork** and **exec**.

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

ERRORS

proc_ctl fails if one or more of the following are true:

[EINVAL] **cmd** is not one of the supported values.

[EPERM] Caller is not root for disabling priority aging, and the binary configuration disallows this.

[EPERM] The caller is not root and the effective uid is not the same as the selected process.

[ESRCH] **who** specifies a non-existent process.

SEE ALSO

nice(2), vm_ctl(2seq)

NOTES

Binary configuration of ability for non-root processes to disable priority aging is provided via a variable in the kernel configuration file *param_space.c*: **root_prio_noage.** If this variable is non-zero, only root can disable priority aging.

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

quotactl – manipulate disk quotas

SYNOPSIS

```
#include <sys/ufsquota.h>

quotactl(cmd, special, uid, addr)
int cmd;
char *special;
int uid;
caddr_t addr;
```

DESCRIPTION

The **quotactl** call manipulates disk quotas. The **cmd** parameter indicates a command to be applied to the user ID **uid**. **special** is a pointer to a null-terminated string containing the pathname of the block special device for the filesystem being manipulated. The block special device must be mounted. **addr** is the address of an optional, command-specific data structure that is copied in or out of the system. The interpretation of **addr** is given with each command below.

Q_QUOTAON
: Turn on quotas for a filesystem. **addr** is a pointer to a null-terminated string containing the pathname of the file containing the quotas for the filesystem. The quota file must exist and the commands for the quota system require that it be named **quotas**. This call is restricted to the superuser.

Q_QUOTAOFF
: Turn off quotas for a filesystem. This call is restricted to the superuser.

Q_GETQUOTA
: Get disk quota limits and current usage for user **uid**. **addr** is a pointer to a struct dqblk structure (defined in *<sys/ufsquota.h>*). Only the superuser may get the quotas of another user.

Q_SETQUOTA
: Set disk quota limits and current usage for user **uid**. **addr** is a pointer to a struct dqblk structure (defined in *<sys/ufsquota.h>*). This call is restricted to the superuser.

Q_SETQLIM
: Set disk quota limits for user **uid**. **addr** is a pointer to a struct dqblk structure (defined in *<sys/ufsquota.h>*). This call is available only to the superuser.

Q_SYNC
: Update the on-disk copy of quota usages. This call is restricted to the superuser. If **special** is NULL, all filesystems with quotas enabled have their on-disk copy of quota usages updated.

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

ERRORS

quotactl fails if any of the following occurs:

[EACCES]	The **addr** parameter is being interpreted as the path of a quota file that exists but is not a regular file or is not on the filesystem pointed to by the **special** parameter.
[EFAULT]	An invalid **addr** is supplied; the associated structure could not be copied in or out of the kernel.
[EINVAL]	**cmd** is invalid.
[ENODEV]	The **special** parameter is not a mounted filesystem.
[ENOTBLK]	The **special** parameter is not a block device.
[EPERM]	The call is privileged and the caller was not the superuser.
[ESRCH]	The quota table is full or the **special** parameter is a mounted filesystem without quotas enabled.

SEE ALSO

quotaon(1M), quotacheck(1M)

NOTES

This is a Sequent library routine. To use it, specify the **-lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

readlink – read value of a symbolic link

SYNOPSIS

int readlink(path, buf, bufsiz)
char *path, *buf;
int bufsiz;

DESCRIPTION

readlink places the contents of the symbolic link **name** in the buffer **buf** which has size **bufsiz**. The contents of the link are not null-terminated when returned.

DIAGNOSTICS

If successful, the count of characters is placed in the buffer. Otherwise a –1 is returned and **errno** is set to indicate the error.

ERRORS

readlink fails if any of the following are true:

[EACCES]	Search permission is denied on a component of the path prefix.
[EFAULT]	**Buf** extends outside the process's allocated address space.
[EINVAL]	The named file is not a symbolic link.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too many symbolic links were encountered in translating the pathname.
[ENOENT]	The pathname was too long.
[ENOENT]	The named file does not exist.
[ENOTDIR]	A component of the path prefix is not a directory.
[ENXIO]	The named file is not a symbolic link.
[EPERM]	The effective user ID does not match the owner of the file and the effective user ID is not the superuser.

SEE ALSO

lstat(2), stat(2), symlink(2seq)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

readv – read vector from file

SYNOPSIS

```
#include <sys/types.h>
#include <sys/uio.h>

int readv(fildes, iov, iovcnt)
int fildes;
struct iovec *iov;
int iovcnt;
```

DESCRIPTION

fildes is a file descriptor obtained from a **creat**(2), **open**(2), **dup**(2), **fcntl**(2), or **pipe**(2) system call.

readv attempts to read data from the file associated with **fildes**, scattering the data into the **iovcnt** buffers specified by the **iovec** array: iov[0], iov[1], ..., iov[iovcnt – 1].

For **readv**, the **iovec** structure is defined as

```
struct iovec {
        caddr_t iov_base;
        int     iov_len;
};
```

On devices capable of seeking, the **readv** starts at a position in the file given by the file pointer associated with **fildes**. Upon return from **readv**, the file pointer is incremented by the number of bytes actually read.

Devices that are incapable of seeking always read from the current position. The value of a file pointer associated with such a file is undefined.

Upon successful completion, **readv** returns the number of bytes actually read and placed in the buffer; this number may be less than the sum of the **iov_len** fields in **iov** if the file is associated with a communication line (see **ioctl**(2) and **termio**(7)), or if the number of bytes left in the file is less than specified by **iov**. A value of 0 is returned when an end-of-file has been reached.

A **readv** from a STREAMS (see **intro**(2)) file can operate in three different modes: *byte-stream* mode, *message-nondiscard* mode, and *message-discard* mode. The default is byte-stream mode. This can be changed using the I_SRDOPT **ioctl** request (see **streamio**(7)), and can be tested with the I_GRDOPT **ioctl**. In byte-stream mode, **readv** retrieves data from the **stream** until it has retrieved enough data to fill the **iov** buffers, or until there is no more data to be retrieved. Byte-stream mode ignores message boundaries.

In STREAMS message-nondiscard mode, **readv** retrieves data until it has read as much data as specified by **iov**, or until it reaches a message boundary. If the **readv** does not retrieve all the data in a message, the remaining data are replaced on the stream, and can be retrieved by the next **readv** or **getmsg**(2) call. Message-discard mode also retrieves data until it has retrieved filled **iov**, or it reaches a message boundary. However, unread data remaining in a message after the **readv** returns are discarded, and are not available for a subsequent **readv** or **getmsg**.

When attempting to read from a regular file with mandatory file/record locking set (see **chmod**(2)), and there is a blocking (i.e., owned by another process) write lock on the segment of the file to be read:

If O_NDELAY is set, the read returns a -1 and sets **errno** to EAGAIN.

If O_NDELAY is clear, the read sleeps until the blocking record lock is removed.

When attempting to read from an empty pipe (or FIFO):

If O_NDELAY is set, the read returns a 0.

If O_NDELAY is clear, the read blocks until data is written to the file or the file is no longer open for writing.

When attempting to read a file associated with a **tty** that has no data currently available:

If O_NDELAY is set, the read returns a 0.

If O_NDELAY is clear, the read blocks until data becomes available.

When attempting to read a file associated with a stream that has no data currently available:

If O_NDELAY is set, the read returns a -1 and sets **errno** to EAGAIN.

If O_NDELAY is clear, the read blocks until data becomes available.

When reading from a STREAMS file, handling of zero-byte messages is determined by the current read mode setting. In byte-stream mode, **readv** accepts data until it has read the number of bytes requested by **iov**, or until there is no more data to read, or until a zero-byte message block is encountered. **readv** then returns the number of bytes read, and places the zero-byte message back on the stream to be retrieved by the next **read**, **readv**, or **getmsg**. In the two other modes, a zero-byte message returns a value of 0 and the message is removed from the stream. When a zero-byte message is read as the first message on a stream, a value of 0 is returned regardless of the read mode.

A **readv** from a STREAMS file can only process data messages. It cannot process any type of protocol message and fails if a protocol message is encountered at the stream head.

DIAGNOSTICS

Upon successful completion, a non-negative integer is returned indicating the number of bytes actually read. Otherwise, a –1 is returned and **errno** is set to indicate the error.

ERRORS

readv fails if any of the following are true:

[EAGAIN]	Mandatory file/record locking was set, O_NDELAY was set, and there was a blocking record lock.
[EAGAIN]	Total amount of system memory available when reading via raw IO is temporarily insufficient.
[EAGAIN]	No message waiting to be read on a stream and O_NDELAY flag set.

[EBADF]	**fildes** is not a valid file descriptor open for reading.
[EBADMSG]	Message waiting to be read on a stream is not a data message.
[EDEADLK]	The read was going to go to sleep and cause a deadlock situation to occur.
[EFAULT]	**iov** or one of the elements in **iov** points outside the allocated address space.
[EINTR]	A signal was caught during the **readv** system call.
[EINVAL]	Attempted to read from a stream linked to a multiplexor.
[EINVAL]	**iovcnt** was less than or equal to 0, or greater than 16.
[EINVAL]	One of the **iov_len** values in the **iov** array was negative.
[EINVAL]	The sum of the **iov_len** values in the **iov** array overflowed a 32-bit integer.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ENOLCK]	The system record lock table was full, so the read could not go to sleep until the blocking record lock was removed.
[ENOMEM]	The size of the read requested from a raw device would have required the process to exceed its maximum resident set size.

A **readv** from a STREAMS file also fails if an error message is received at the stream head. In this case, **errno** is set to the value returned in the error message. If a hangup occurs on the stream being read, **readv** continues to operate normally until the stream head read queue is empty. Thereafter, it returns 0.

SEE ALSO

creat(2), dup(2), fcntl(2), ioctl(2), intro(2), open(2), pipe(2), getmsg(2), read(2), writev(2), streamio(7), termio(7)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

setdtablesize – set descriptor table size

SYNOPSIS

setdtablesize(size)
int size;

DESCRIPTION

This function sets the descriptor table size in the calling process to at least the minimum of the **size** argument and a system-defined maximum number of file descriptors (binary-configurable). The implementation can grant more file descriptors than the number requested. **setdtablesize** returns the number of file descriptors the process can now use. A request to decrease the number of file descriptors in a process is honored by closing all open file descriptors with indices equal to or greater than **size**.

The maximum number of file descriptors is guaranteed to be at least 20.

Before **setdtablesize** can be used to increase the number of file descriptors, NOFILES and NOFILEEXT must be set as options in the binary configuration file.

The new descriptor table size is inherited across **fork**(2) and kept across **exec**(2) system calls.

DIAGNOSTICS

If successful, the number of usable file descriptors in the calling process is returned. Otherwise, a –1 is returned, and the global variable **errno** is set to indicate the error.

ERRORS

[EAGAIN]	The system was temporarily unable to allocate sufficient memory resources to satisfy the request.
[EINVAL]	The *size* argument was invalid (*size* is less than zero).
[ENFILE]	The system had no free open-file table extension objects.

SEE ALSO

intro(2), close(2), dup(2), getdtablesize(2seq), open(2)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

setgroups – set group access list

SYNOPSIS

#include <sys/types.h>

setgroups(gidsetsize, grouplist)
int gidsetsize;
gid_t grouplist[];

DESCRIPTION

setgroups sets the group access list of the current user process according to the array **grouplist**. The parameter **gidsetsize** indicates the number of entries in the array and must be no more than NGROUPS_MAX, as defined in *<limits.h>*.

Only the superuser may set new groups.

DIAGNOSTICS

A 0 value is returned on success, –1 on error with **errno** set to indicate the error.

ERRORS

setgroups fails if:

[EFAULT]	The address specified for **grouplist** is outside the process address space.
[EINVAL]	The size of group array is greater than NGROUPS_MAX.
[EPERM]	The caller is not the superuser.

SEE ALSO

getgroups(2)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

sigcontext – get signal context

SYNOPSIS

```
#include <signal.h>

sigcontext(req, addr);
int req;
int *addr;
```

DESCRIPTION

sigcontext allows users to determine the context of the signal being delivered to a signal handler. The data, which can be returned or set to the location specified by **addr**, can be requested from within a signal handler by setting **req** to one of the following:

SF_SIGNO
: requests the current signal number.

SF_CODE
: requests the *code* value. This is set by SIGILL, SIGFPE, SIGSEGV, and SIGPWR (on SSM based systems) signals to indicate the reason for the signal on an exception (see *signal.h* and *fpu.h*). For all other reasons (including use of **kill**(2)), the value -1 is returned.

SF_SYSCODE
: Indicates the system call number if the signal was taken during a system call (see *syscall.h* and *sysseq.h*), otherwise returns -1.

SF_TRAPCODE
: Indicates the machine trap number if the signal was taken during a trap (see *trap.h*), otherwise returns -1.

SF_SEGVCODE
: requests the illegal address in the case of SIGSEGV. For all other signals this value is undefined.

SF_PC
: requests the value of the program counter at the time the signal was delivered.

SF_SP
: requests the value of the stack pointer at the time the signal was delivered.

SF_R0
: requests the value of the first temporary register at the time the signal was delivered.

SF_CODE_DATA
: requests the additional data set by some of the signals (such as SIGFPE). This is only valid if one of the SF_CODE fields is set (which should be checked first).

SF_SET_PC
: sets the return program counter to be **addr.**

The information requested relates to the last signal caught from a signal handler, and therefore nested signal handlers may request information about the signal that caused the handler to be invoked. The vaue is undefined when the

data is called outside of a signal handler or after a **longjmp(3)**.

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

ERRORS

sigcontext fails if one of the following occurs:

[EFAULT]	**addr** points to memory which is not a valid part of the process address space, or no signal frame exists.
[EINVAL]	**req** is not valid.

SEE ALSO

sigaction(2), sigstack(2seq), setjmp(3)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

sigstack – set and/or get signal stack context

SYNOPSIS

```
#include <signal.h>

struct sigstack {
        caddr_t         ss_sp;
        int     ss_onstack;
};

sigstack(ss, oss);
struct sigstack *ss, *oss;
```

DESCRIPTION

sigstack allows users to define an alternative stack on which signals are to be processed. If **ss** is non-zero, it specifies a **signal stack** on which to deliver signals and tells the system whether the process is currently executing on that stack. When a signal's action indicates its handler should execute on the signal stack (specified with a **sigaction**(2) call), the system checks to see if the process is currently executing on that stack. If the process is not currently executing on the signal stack, the system arranges a switch to the signal stack for the duration of the signal handler's execution. If **oss** is non-zero, the current signal stack state is returned. **ss_onstack** is zero when executing on the normal stack, otherwise it is non-zero.

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

ERRORS

sigstack fails and the signal stack context remains unchanged if the following occurs:

[EFAULT] Either **ss** or **oss** points to memory that is not a valid part of the process address space.

SEE ALSO

sigaction(2), setjmp(3)

NOTES

Signal stacks are not grown automatically, as is done for the normal stack. If the stack overflows, unpredictable results may occur.

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

symlink – make symbolic link to a file

SYNOPSIS

symlink(name1, name2)
char *name1, *name2;

DESCRIPTION

A symbolic link **name2** is created to **name1** (**name2** is the name of the file created, **name1** is the string used in creating the symbolic link). Either name may be an arbitrary pathname; the files need not be on the same file system.

DIAGNOSTICS

Upon successful completion, a 0 is returned. Otherwise, –1 value is returned and **errno** is set to indicate the error.

ERRORS

The symbolic link is made unless one or more of the following are true:

[EACCES]	A component of the **name2** path prefix denies search permission.
[EEXIST]	**name2** already exists.
[EFAULT]	**name1** or **name2** points outside the process's allocated address space.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[ELOOP]	Too may symbolic links were encountered in translating the pathname.
[ENOENT]	One of the pathnames specified was too long.
[ENOSPC]	The directory in which the entry for the new symbolic link is being placed cannot be extended because there is no space left on the file system containing the directory.
[ENOSPC]	The new symbolic link cannot be created because there is no space left on the file system which will contain the link.
[ENOSPC]	There are no free inodes on the file system on which the symbolic link is being created.
[ENOTDIR]	A component of the **name2** prefix is not a directory.
[EROFS]	The file **name2** would reside on a read-only file system.

SEE ALSO

link(2), ln(1), lstat(2), readlink(2seq), unlink(2)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

tmp_affinity – bind a process to a processor

SYNOPSIS

```
#include <sys/tmp_ctl.h>

int tmp_affinity(processor);
int  processor;
```

DESCRIPTION

tmp_affinity allows a process to be bound to a specified logical processor. Processor numbers start at zero and are numbered contiguously. Deconfigured processors are not included. The previous affinity is returned.

If the **processor** argument is **AFF_NONE**, the process is realeased from any previous affinity (that is, allowed to migrate). If the **processor** argument is **AFF_QUERY**, the current affinity, or **AFF_NONE** is returned, without changing the current affinity.

Only the superuser is allowed to change affinity.

Affinity is inherited across the **fork**(2) and **exec**(2) system calls.

DIAGNOSTICS

If successful, either the previous affinity processor number or AFF_NONE is returned. On error, –1 is returned and **errno** is set to indicate the error.

ERRORS

The **tmp_affinity** fails if:

[ENODEV]	**processor** is either offline or bad.
[ENXIO]	**processor** is out of range.
[EPERM]	Process does not have superuser capability.

SEE ALSO

tmp_ctl(2seq), online(1M)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

tmp_ctl – multi-processor control

SYNOPSIS

#include <sys/tmp_ctl.h>

int tmp_ctl(command, processor)
int command, processor;

DESCRIPTION

tmp_ctl allows processes to query the status of the processor pool resource. Superuser processes can also change the configuration. The defined commands (from **tmp_ctl.h**) have the following meanings:

TMP_NENG
: returns the number of processors configured in the processor pool. The processors may be online or offline. However, deconfigured processors will not be included. The **processor** argument is ignored.

TMP_OFFLINE
: shuts down a processor. The processor stops dispatching processes and is paused. Only the superuser is allowed to offline a processor.

TMP_ONLINE
: starts up a processor. The processor initializes and begins scheduling user processes. Only the superuser is allowed to online a processor.

TMP_QUERY
: returns the state (TMP_ENG_ONLINE or TMP_ENG_OFFLINE) of the selected processor.

DIGANOSTICS

The call returns 0 on successful completion, or the desired information in the case of the query operations. The call returns –1 on error and **errno** is set to indicate the error.

ERRORS

The **tmp_ctl** fails if:

[EBUSY]	Attempting to offline a processor that has bound drivers or bound processes, or attempting to offline the last processor.
[EINVAL]	**processor** is already offline or online.
[ENODEV]	**processor** is bad.
[ENXIO]	**processor** is out of range.
[EPERM]	Process does not have superuser capability and an attempt to change the processor pool configuration was made.

SEE ALSO

tmp_affinity(2seq), online(1M)

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

truncate, ftruncate – truncate a file to a specified length

SYNOPSIS

```
#include <sys/types.h>

truncate(path, length)
char *path;
off_t length;

ftruncate(fd, length)
int fd;
off_t length;
```

DESCRIPTION

truncate causes the file named by **path** or referenced by **fd** to be truncated to at most **length** bytes in size. If the file previously was larger than this size, the extra data is lost. With **ftruncate**, the file must be open for writing. **truncate** and **ftruncate** both interact with mandatory record locking. If the file being truncated has locks asserted in the portion of the file being removed, **truncate** and **ftruncate** block until the overlapping locks are removed. In the case of **ftruncate**, if the **fd** has the O_NDELAY flag set, the system call fails.

DIAGNOSTICS

A value of 0 is returned if the call succeeds. If the call fails, a –1 is returned, and **errno** is set to indicate the error.

ERRORS

truncate fails if any of the following is true:

[EACCES]	A component of the **path** prefix denies search permission.
[EAGAIN]	**path** references a file with mandatory record locking enabled, and the portion of the file being deleted overlaps a record lock.
[EFAULT]	**path** points outside the process's allocated address space.
[EISDIR]	The named file is a directory.
[ENOENT]	The pathname was too long.
[ENOENT]	The named file does not exist.
[ENOTDIR]	A component of the path prefix of **path** is not a directory.
[EROFS]	The named file resides on a read-only file system.
[ETXTBSY]	The file is a pure procedure (shared text) file that is being executed.

ftruncate fails if any of the following are true:

[AGAIN]	**fd** references a file with mandatory record locking enabled, and the portion of the file being deleted overlaps a record lock.
[EBADF]	The **fd** is not a valid descriptor.
[EIO]	An I/O error occurred while reading from or writing to the file system.

SEE ALSO

open(2), fcntl(2)

NOTES

These are Sequent library routines. To use them, specify the **-lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

BUGS

Partial blocks discarded as the result of truncation are not zero filled. This can result in holes in files which do not read as zeros.

NAME

vm_ctl – examine and change virtual memory tuning parameters

SYNOPSIS

```
#include <sys/types.h>
#include <sys/vmsystm.h>

vm_ctl(operation, argp)
int operation;
union {
        struct vm_tune *vmtune;
        unsigned long *procrss;
        bool_t onoff;
}       argp;
```

DESCRIPTION

vm_ctl examines and changes virtual memory (VM) related parameters. The function performed depends on the value of **operation**.

If **operation** is VM_GETPARAM, **vm_ctl** fetches the system-wide VM tuning parameters into the structure pointed to by **vmtune**. No special permissions are required to obtain this information.

If **operation** is VM_SETPARAM, the system-wide VM tuning parameters become the values in the structure pointed to by **vmtune**. Existing processes are unaffected by the per-process tuning parameters (such as minimum resident set size); new processes, however, are bound by the new values. This form of the call is available only to the superuser.

If **operation** is VM_GETRS, the unsigned long pointed to by **procrss** is filled with the resident set size of the calling process. No special permissions are required to obtain this information.

If **operation** is VM_SETRS, the resident set size of the calling process is set to the unsigned long pointed to by **procrss**. This value must be within the system-wide bounds currently enforced via the VM_SETPARAM operation (described above), and the per-process resident set size restriction. This form of the call is available only to the superuser.

If **operation** is VM_SWAPABLE, the calling process is declared non-swappable if onoff is zero; otherwise, the calling process is declared swappable. The swappable attribute is inherited across **fork**s and **exec**s. This form of the call is available only to the superuser, unless the proper configuration parameter has been altered (see **NOTES** later). Notice that VM_SWAPABLE has only one *P*.

If **operation** is VM_PFFABLE, the calling process is declared non-PFFable if onoff is zero; otherwise, the calling process is declared PFFable. When a process is PFFable, its resident set size is altered based on the frequency at which the process page-faults. The PFFable attribute is inherited across **fork**s and **exec**s. This form of the call is available only to the superuser, unless the proper configuration parameter has been altered (see NOTES later).

DIAGNOSTICS

If the call succeeds, 0 is returned. Otherwise, a –1 is returned, and **errno** is set to indicate the error.

The structure pointed to by **vmtune** is defined in *<sys/vmsystm.h>* as:

```
/*
 * VM System parameters; tunable.
 *
 * Where appropriate, the units (HW pages or clusters)
 * are noted.
 * PFF rate units (PFFlow, PFFhigh) may change if
 * faults/sec isn't enough resolution.
 */

structvm_tune     {
  longvt_minRS;    /*min # clusters for Rset*/
  longvt_maxRS;    /*max # clusters for Rset;limited by maxRS*/
  longvt_RSexecslop;     /*# HW pages slop in exec*/
  longvt_RSexecmult;     /*Rset size multipler*/
  longvt_RSexecdiv;      /*Rset size divider*/
  int vt_dirtylow;/*dirty-list low-water mark (HW pgs)*/
  int vt_dirtyhigh;      /*dirty-list high-water mark (HW pgs)*/
  int vt_klout_look;     /*pageout kluster look-ahead (clusters)*/
  longvt_PFFvtime;/*ticks between PFF adjusts; 0=>no PFF*/
  longvt_PFFdecr; /*# clusters to decrement if PFF<PFFlow*/
  longvt_PFFlow;  /*low PFF rate, <=PFFhigh (faults/sec)*/
  longvt_PFFincr; /*# clusters to add if PFF > PFFhigh*/
  longvt_PFFhigh; /*high PFF rate (faults/sec)*/
  longvt_minfree; /*freelist low bound for swap (HW pgs)*/
  longvt_desfree; /*freelist high bound for swap (HW pgs)*/
  longvt_maxdirty;/*max dirty-list before swap (HW pgs)*/
};
```

ERRORS

vm_ctl fails if any of the following is true:

[EFAULT] The memory pointed to by **vmtune** or **procrss** lies outside the valid address space for the process.

[EINVAL] **operation** is not one of the supported values.

[EINVAL] An element of the structure pointed to by **vmtune** is not within permissible range.

[EPERM] The process does not have the proper permissions to make the requested change.

SEE ALSO

get_process_stats(2seq), mmap(2seq), munmap(2seq), plock(2), vmtune(1M)

NOTES

Setting the resident set size (with VM_SETRS) does not actually change the current number of resident pages. Pages are added or removed through normal page faulting activity.

The system verifies that the tuning parameters at least have some degree of sanity to avoid immediate obvious panics. However, ignorant alterations of tuning parameters will most certainly result in a severe degradation in

response.

Programs should not mix calls to **vm_ctl** (only the VM_SWAPABLE case) and **plock**(2). See the **NOTES** section of **plock**(2).

Binary configuration of ability for non-root processes to disable swapping, PFF-ing, or adjusting the size of the resident set is provided via three variables in the kernel configuration file *param_space.c*: **root_vm_swapable**, **root_vm_pffable**, **root_vm_setrs**, respectively. If any of these variables is non-zero, the corresponding function is available only to root processes. Notice that **root_vm_swapable** has only one *P*.

This is a Sequent library routine. To use it, specify the **-lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.

NAME

writev – write vector on a file

SYNOPSIS

```
#include <sys/types.h>
#include <sys/uio.h>

writev(d, iov, ioveclen)
int d;
struct iovec *iov;
int ioveclen;
```

DESCRIPTION

fildes is a file descriptor obtained from a **creat**(2), **open**(2), **dup**(2), **fcntl**(2), or **pipe**(2) system call.

writev attempts to write to the file associated with the **fildes**, data gathered from the **iovlen** buffers specified by the members of the **iovec** array: **iov[0]**, **iov[1]**, etc.

On devices capable of seeking, the actual writing of data proceeds from the position in the file indicated by the file pointer. Upon return from **writev**, the file pointer is incremented by the number of bytes actually written.

On devices incapable of seeking, writing always takes place starting at the current position. The value of a file pointer associated with such a device is undefined.

If the O_APPEND flag of the file status flags is set, the file pointer is set to the end of the file prior to each write.

For regular files, if the O_SYNC flag of the file status flags is set, the write will not return until both the file data and file status have been physically updated. This function is for special applications that require extra reliability at the cost of performance. For block special files, if O_SYNC is set, the write will not return until the data has been physically updated.

A write to a regular file is blocked if mandatory file/record locking is set (see **chmod**(2)), and there is a record lock owned by another process on the segment of the file to be written. If O_NDELAY is not set, the write sleeps until the blocking record lock is removed.

Let **nbyte** represent the sum of all the buffers specified by **iov**. For STREAMS (see **intro**(2)) files, the operation of **writev** is determined by the values of the minimum and maximum **nbyte** range (packet size) accepted by the stream. These values are contained in the topmost stream module. Unless the user pushes (see I_PUSH in **streamio**(7)) the topmost module, these values can not be set or tested from user level. If **nbyte** falls within the packet size range, **nbyte** bytes are written. If **nbyte** does not fall within the packet size range and the minimum packet size value is zero, **writev** breaks the buffer into maximum packet size segments prior to sending the data downstream (the last segment may contain less than the maximum packet size). If **nbyte** does not fall within the packet size range and the minimum value is non-zero, **writev** fails with **errno** set to ERANGE. Writing a zero-length buffer (**nbyte** is zero) sends zero bytes with zero returned.

For STREAMS files, if O_NDELAY is not set and the stream can not accept data (the stream write queue is full due to internal flow control conditions), **writev** blocks until data can be accepted. O_NDELAY prevents a process from blocking due to flow control conditions. If O_NDELAY is set and the stream can not accept data, **writev** fails. If O_NDELAY is set and part of the buffer has been written when a condition in which the stream can not accept additional data occurs, **writev** terminates and returns the number of bytes written.

writev fails and the file pointer remains unchanged if one or more of the following are true:

[EAGAIN] Mandatory file/record locking was set, O_NDELAY was set, and there was a blocking record lock.

[EAGAIN] Total amount of system memory available when reading via raw I/O is temporarily insufficient.

[EAGAIN] Attempt to write to a stream that can not accept data with the O_NDELAY flag set.

[EBADF] **fildes** is not a valid file descriptor open for writing.

[EDEADLK] The write was going to go to sleep and cause a deadlock situation to occur.

[EFAULT] **iov** or one of its members points outside the process's allocated address space.

[EFBIG] An attempt was made to write a file that exceeds the process's file size limit or the maximum file size (see **ulimit**(2)).

[EINTR] A signal was caught during the **writev** system call.

[EINVAL] Attempt to write to a stream linked below a multiplexor.

[ENOLCK] The system record lock table was full, so the write could not go to sleep until the blocking record lock was removed.

[ENOSPC] During a **writev** to an ordinary file, there is no free space left on the device.

[ENXIO] A hangup occurred on the stream being written to.

[EPIPE and SIGPIPE signal] An attempt is made to write to a pipe that is not open for reading by any process.

[ERANGE] Attempt to write to a stream with number of bytes in the request outside specified minimum and maximum write range, and the minimum value is non-zero.

[ENOMEM] The size of the write request to a raw device would have required the process to exceed its maximum resident set size.

[EIO] A hangup occurred on the stream being written to.

[EIO] An I/O error occurred while reading from or writing to the file system.

[EIO] The process is in a background process group and is attempting to write to its controlling terminal. TGSTOP is set, the process is neither ignoring nor blocking SIGTTOU signals, and the

process group of the process is orphaned.

[EINVAL] **iovcnt** was less than or equal to 0, or greater than 16.

[EINVAL] One of the **iov_len** values in the **iov** array was negative.

[EINVAL] The sum of the **iov_len** values in the **iov** array overflowed a 32-bit integer.

If a **writev** requests that more bytes be written than there is room for (e.g., the **ulimit** (see **ulimit**(2)) or the physical end of a medium), only as many bytes as there is room for are written. For example, suppose there is space for 20 bytes more in a file before reaching a limit. A write of 512-bytes returns 20. The next write of a non-zero number of bytes gives a failure return (except as noted below).

If the file being written is a pipe (or FIFO) and the O_NDELAY flag of the file flag word is set, then write to a full pipe (or FIFO) returns a count of 0. Otherwise (O_NDELAY clear), writes to a full pipe (or FIFO) blocks until space becomes available.

A write to a STREAMS file can fail if an error message has been received at the stream head. In this case, **errno** is set to the value included in the error message.

SEE ALSO

creat(2), dup(2), fcntl(2), intro(2), lseek(2), open(2), pipe(2), readv(2seq), ulimit(2), write(2)

DIAGNOSTICS

Upon successful completion, the number of bytes actually written is returned. Otherwise, –1 is returned and **errno** is set to indicate the error.

NOTES

This is a Sequent library routine. To use it, specify the **–lseq** option to link with the Sequent library. Since this is a Sequent extension, code using this routine may not be portable to other systems.